An object-oriented database for the compilation of signal transduction pathways

Von der Gemeinsamen Naturwissenschaftlichen Fakultät
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigte

D i s s e r t a t i o n

von
Frank Schacherer
aus Freiburg

1. Referent: Priv. Doz. Dr. U. Bilitewski
2. Referent: Prof. Dr. H. D. Ehrich
eingereicht am: 8.3.2001
mündliche Prüfung am: 3.7.2001

2001
(Druckjahr)

Vorveröffentlichungen der Dissertation

Teilergebnisse aus dieser Arbeit wurden mit Genehmigung der gemeinsamen Naturwissenschaftlichen Fakultät, vertreten durch die Mentorin oder den Mentor/die Betreuerin oder den Betreuer der Arbeit, in folgenden Beiträgen vorab veröffentlicht:

Publikationen

Wingender, E., Chen, X., Fricke, E., Geffers, R., Hehl, R., Liebich, I., Krull, M., Matys, V., Michael, H., Ohnhuser, R., Prüß, M., Schacherer, F., Thiele, S., Urbach, S. The TRANSFAC system on gene expression regulation. Nucleic Acids Res. 29 (2001) in press

Wingender, E., Chen, X., Hehl, R., Karas, H., Liebich, I., Matys, V., Meinhardt, T.,Prüß, M.., Reuter, I. and Schacherer, F. TRANSFAC: an integrated system for gene expression regulation. Nucleic Acids Res. 28, 316–319 (2000)

Heinemeyer, T., Chen, X., Karas, H., Kel, A. E., Kel, O. V., Liebich, I., Meinhardt, T., Reuter, I., Schacherer, F. and Wingender, E. Expanding the TRANSFAC database towards an expert system of regulatory molecular mechanisms. Nucleic Acids Res. 27, 318–322 (1999).

Tagungsbeiträge

Schacherer, F., Wingender E. The TRANSPATH Signal Transduction Database: A knowledge base on signal transduction networks. (Poster) Intelligent Systems in Molecular Biology (2000).

Schacherer, F., Choi, C., Götze, U., Krull, M., Wingender E., The TRANSPATH Signal Transduction Database: A knowledge base on signal transduction networks (Poster) Novosibirsk (2000).

Schacherer, F., Wingender E. The Transpath Signal Transduction Database. (Poster) German Conference on Bioinformatics (1999).

Schacherer, F., Wingender E. The Transpath Signal Transduction Database. (Poster) Conference of the Signal Transduiction Society (1999).

"*If the Lord Almighty had consulted me*
before embarking on creation,
I should have recommended something simpler."

— Alphonso X (1221–1248)
(Alphonso the Wise)
King of Castile and Leon

Contents

1 INTRODUCTION **3**

1.1 Motivation . . . 3
1.2 Biological signal transduction . . . 6
1.2.1 Signal hierarchies . . . 6
1.2.2 Signaling motifs . . . 7
1.2.3 Extracellular signaling . . . 8
1.2.4 Receptors . . . 8
1.2.5 Intracellular signaling . . . 9
1.2.6 Nature of the cytosol . . . 10
1.3 Object-Oriented Databases . . . 10

2 RESULTS **13**

2.1 Data model for the signaling network . . . 13
2.2 Querying the signaling network . . . 13
2.3 Visualising the signaling network . . . 16
2.4 Class layout for the Transpath database . . . 16
2.4.1 Entry . . . 19
2.4.2 Component . . . 19
2.4.3 Molecule . . . 19
2.4.4 Motif . . . 20
2.4.5 Family . . . 20
2.4.6 State . . . 22
2.4.7 Reaction . . . 23
2.4.8 Connection . . . 27
2.4.9 Location . . . 27
2.4.10 Hyperlink, Annotate, Reference . . . 28
2.4.11 Method, Material . . . 28
2.4.12 Signal . . . 28
2.4.13 Pathway . . . 29
2.5 Interface for the Transpath database . . . 29
2.5.1 Sessions . . . 29
2.5.2 Data entry with a form client . . . 31
2.5.3 Web navigation interface . . . 32
2.5.4 Graphical Pathway Builder . . . 32
2.5.5 Flatfile generation . . . 40
2.5.6 Parsers and data import . . . 40

3 APPLICATIONS **43**
3.1 Path alignments . . . 43
3.1.1 Paths . . . 44
3.1.2 Assumptions . . . 44
3.1.3 Method . . . 45
3.1.4 Results . . . 46
3.2 Qualitative simulation with Boolean Networks . . . 48
3.2.1 Boolean networks . . . 48
3.2.2 Modeling assumptions . . . 48
3.2.3 Modeling procedure and parameters . . . 49
3.2.4 Results . . . 49
3.3 Quantitative simulation with Petri Nets . . . 53
3.3.1 Petri nets . . . 53
3.3.2 Modeling assumptions . . . 53
3.3.3 Modeling procedure and parameters . . . 56
3.3.4 Results . . . 57

4 DISCUSSION **65**
4.1 Data . . . 65
4.2 Database . . . 66
4.2.1 Graph representations . . . 66
4.2.2 DBMS . . . 67
4.2.3 Visualisation . . . 67
4.2.4 Class layout . . . 69
4.2.5 Interface . . . 72
4.3 Previous work . . . 73
4.3.1 Databases . . . 73
4.3.2 Simulations . . . 74
4.4 Applications . . . 77
4.4.1 Path alignment . . . 78
4.4.2 Boolean networks . . . 79
4.4.3 Petri nets . . . 80
4.5 Status and future of the database . . . 81
4.6 Open questions . . . 81
4.7 Summary . . . 83

A MATERIALS AND METHODS **91**

B ABBREVIATIONS **95**

C PATH ALIGNMENT **97**

D PETRI NET PARAMETERS **101**

E ADMINISTRATIVA AND THANKS **107**

Chapter 1

INTRODUCTION

It is the basic tenet of research in the natural sciences that theory has to be validated by experimental data[Fey67]. In an iterative process, experiment and observation yield data, which is then organized and becomes theory. The theory in turn leads to new experiments.

Currently the field of biology in the main and of signal transduction in particular is preoccupied with data-collection, and the overall signaling system is still unknown. What we have is a hodge-podge of individual observations. The situation is akin to the one chemists faced in the eighteenth century: the amount of data generated was unwieldy and it was difficult to make sense of it. It was not before the periodic table was created, which was based on that raw data and made a system evident in those observations, that a model could be developed to explain the multitude of phenomena by a set of general rules. By analogy, the data generated today by biologists is collected in structured repositories, and the hope is that one day, based on this data, it will be possible to explain the logic of signal transduction, gene expression patterns, development and cancer.

Physics makes use of mathematics to develop its models, and while the mathematics used is often straightforward from a mathematician's point of view, from the marriage between the two disciplines new insight emerges. Biology deals with systems more complicated and messy than those of physics, and this has made useful mathematical formalization of biological systems difficult to date. Instead, computer databases are used to integrate the knowledge, and while the informatics used are straightforward from an informatician's point of view, from the marriage between the two disciplines a new way of dealing with biological data and doing research has emerged.

1.1 Motivation

Cells, especially those of a complex multicellular organism, have to react to each other and to external influences in a well-concerted manner. If we want to understand cellular behavior and its responses to external signals, or want to influence it in a predictable manner, we have to understand the pathways through which these signals are mediated into and within the cell. Biological signaling pathways also interact with each other to form complex networks. These networks show emergent properties like signal integration or self-sustaining feedback loops

which are not present in the isolated pathways [BI99].

In most cases changes in cell behavior involve the execution of transcriptional events, which are specific for each signal in its cellular context [HT95]. Signal transduction pathways regulate the activity of many transcription factors [Mon97]. The transcription factors in turn are organized in gene-regulatory networks, and the genes expressed can also feed back into the signal transduction network.

Knowledge about the principle mechanisms of signal transduction and regulation mechanisms of individual macromolecules in signaling pathways has multiplied in the last decade. It is now growing at a rate that makes it difficult to keep up with [Kra97]. The huge and ever more rapidly growing amount of signal transduction data demands a database that stores and organizes this knowledge, providing simple and fast access to the information. The complexity created by the crosstalk between pathways makes it virtually impossible to infer by hand all the consequences that follow after one modifies one part of the network. To this end, computer-aided analysis and simulation will have to be used. It can only be successful on the basis of a comprehensive and detailed dataset.

The main purpose of this work is the design of a database for the available data on signal transduction. This database should be usable for two purposes: to provide an information resource for researchers interested in signal transduction and to serve as the basis for automated analysis and simulation work.

Today, a big part of our knowledge about signal transduction is still of a qualitative, indirect nature. The knowledge is fragmented in thousands of single observations, and it is a time-consuming and difficult process to get an overview about it from reading primary literature.

In scientific review papers, graphics like Fig. 1.1 are published to provide an integrated view of the signal pathways.

Analyses of cellular function, termed functional genomics and functional proteomics will be the next big challenge for the whole field of biosciences in the years to come. The function of a molecule has no meaning on its own. Only context gives meaning to the function, and so we try to understand the interactions of each molecule with other molecules. Among many other things, data on signal transduction will be a necessity for this to succeed.

Cancer is the pathological deregulation of cell growth, resulting in uncontrolled proliferation. Practically all oncogenes encode aberrantly functioning members of signal transduction pathways coupled to growth-regulating signals [EW93] that then constitutively switch on transcription. It has been shown[HW00] that a multitude of genes can play a role in oncogenic transformation of cells and that mutation in a single gene alone is not sufficient to induce cancer. Having information about alternate pathways and the interaction between pathways will help to understand abnormal behavior and to target critical elements in pathways for medical treatment.

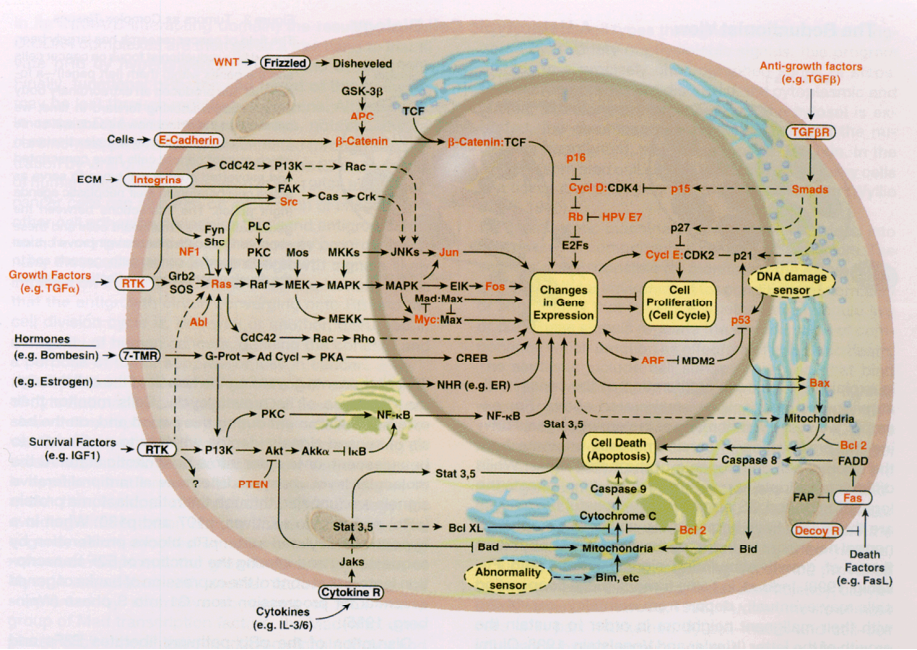

Figure 1.1: Network representation in review literature. Depiction of the cell signaling network in respect of cancer.[HW00] Point arrows are activating reactions, bar-ended arrows are inhibiting ones. Inhibiting arrows in some cases are shown to act on molecules, in other cases they act on reactions.

Keyword	Number in human	Number over all species
SIGNAL	1551	18161
HORMONE	58	953
GROWTH FACTOR	105	589
NEUROTRANSMITTER	30	162
IONIC CHANNEL	147	1357
RECEPTOR	279	4565
G-PROTEIN	227	1357
KINASE	74	4425
TRANSCRIPTION	515	5761
ALL	2986	37330

Table 1.1: Number of signaling proteins found in the SWISS-PROT protein database, public release Nr. 36. Proteins in SWISS-PROT are labeled with keywords. For example, 1551 entries for human were labeled with the keyword SIGNAL.

1.2 Biological signal transduction

What follows is a short outline of well-studied signal transduction principles, molecules and pathways. It will become clear that the means of signal transduction are nearly as varied as the number of pathways.

Biological signaling is performed by the interaction between signaling molecules, and most of them are proteins. Table 1.1 gives an overview of the amount of signaling proteins found in the SWISS-PROT database [BA00]. A lot of the annotation in this database is by homology to experimentally confirmed results. Many pathways that are known today have been assembled from experiments in different model organisms, and checked for humans only with sequence homology comparison of the genes involved. Especially with vertebrates, all major pathways seem to be strongly conserved [OG98].

Judging from the 5908 entries that are in SWISS-PROT for human, compared to the estimated thirty thousand human genes [V$^+$01], this number is one order of magnitude too small. We can expect more than ten thousand proteins to be involved in signal transduction and gene regulation in humans.

1.2.1 Signal hierarchies

Multicellular organisms are highly complex systems. To handle this complexity they are built in a hierarchical manner, and signaling networks exist on several levels of this hierarchy. On the highest level, intercellular signaling via hormones and other factors coordinates the activities of the different organs and tissues. Then, inside the cell the signal transduction network translates the hormonal inputs into outputs that may result in a changed state of the cell or the secretion of other messengers. Inside the cellular nucleus, the transduced signal in turn serves as input for a gene regulatory network and may lead to the expression of other genes as output to the signaling network.

To insulate the different networks from each other, several mechanisms exist. The most important one is spatial compartmentalization: the various networks of signaling molecules exist in spaces which are separated by membranes like

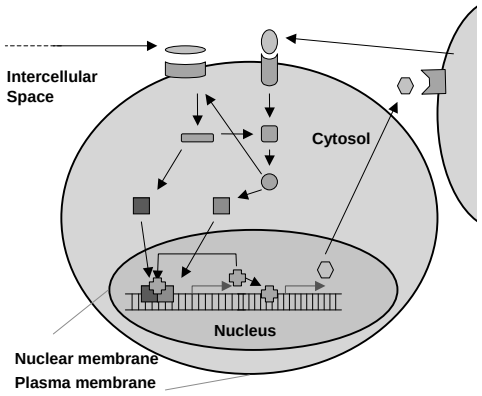

Figure 1.2: Compartimental hierarchies. Signal transduction connects gene-regulatory and intercellular networks.

the plasma or nuclear membrane (Fig.1.2). They interact only at well defined and strongly regulated ports, like receptor or channel proteins.[1]

Another insulation mechanism may be time-scale decoupling [RCPÖ99]. Times of processes in the cell vary significantly from seconds in enzymatic catalysis or even femtoseconds in photosynthetic excitation to dozens of minutes for protein expression and even days or weeks for hormone responses. Slower processes are insensitive to the values of fast-changing processes, seeing only their averaged value.

While it is clear that to fully understand cellular behavior in multicellular organisms, it will be necessary to include the inter-cellular signaling relations in the model, our focus is on intracellular signalling.

1.2.2 Signaling motifs

The molecule is not the smallest unit that interacts in signaling networks. Many proteins are large and have themselves a modular composition, consisting of several *domains*. The domains in turn can feature *motifs*, specific three dimensional structures. Motifs are responsible for the signaling properties and work somehow independently from the rest of the protein [CRB95, LFSSC93].

Compartmentalization does not stop at the subcellular compartment level. Eukaryotic cells have evolved scaffolding and adapter proteins using motifs to simultaneously bind multiple components of a signaling pathway and thereby impose a degree of specificity and order on the highly connected network of signaling molecules.

Scaffolding insulates these molecules and may diminish cross-talk to other pathways. It makes possible a new degree of signal amplification and speed, which could not be achieved with molecules freely distributed in the cytoplasm and relying on diffusion to collide and interact every now and then.

[1] This can be compared to the use of modules and composition in programming. Object-oriented programming incorporates ideas seen in biological systems to handle complexity [Boo94]: you have subsystems with strong interactions or coupling within, but weaker interactions or coupling between subsystems, over well defined interfaces. The overall system is more robust, as changes in one subsystem do not influence the others strongly.

Domains can be shuffled between proteins by evolutionary genetic mechanisms, leading to new composites that may integrate signals in a new way.

1.2.3 Extracellular signaling

Extracellular molecules are recognized by specific receptors on the surface of or within the cell. Since receptors have to detect low concentrations of signal molecules (nanomolar or smaller), they are highly sensitive.

Different kinds of cells have different sets of receptors, and thus can react to different signals. Several signals can be integrated in the cell into a combined signal, and so the combination of a limited amount of messengers can create a vast number of messages.

Different kinds of cells can react differently to the same signal, because they have different interior signal transduction pathways hooked up to the same receptor.

For example acetylcholine makes skeletal muscle contract, but heart muscle decontract, as they use different receptors. It makes secretory cells secrete, even though these have the same receptor as heart muscle, because in this case that same receptor is hooked up to different pathways.

Extracellular signals can be classified according to their chemical composition, to their method of action or to the range and time over which they transfer the signal. The most important ones are hormones and cell surface molecules.

Cell surface molecules are proteins, and many are glycosylated. They are presented on the surface of cells and bind to receptors on the surface of neighboring cells, acting *locally*.

Local regulators are molecules that also act *locally* but are soluble and diffuse through the intercellular space. Some examples are neurotransmitters in synapses, interleukines and nitric oxide, a gas that diffuses through the cell membrane.

Hormones are chemical signals that diffuse *systemically*, carried by blood and lymph. They can broadly be sub-categorized into

1. Hydrophilic molecules like proteins, small peptides, nucleotides, amino acids or their derivatives. They cannot cross the plasma membrane and have to bind to receptors on the cell surface. Usually they have a half-life in blood of several minutes. Short-lived ones also serve as local regulators.
2. Small hydrophobic molecules like steroid hormones, thyroid hormones and retinoids. Bound to carrier molecules, they can travel through the blood stream for hours, thyroid hormones even for days. Finally, they diffuse though the plasma membrane of their target cells, and bind to intracellular receptors.

1.2.4 Receptors

Receptors provide the interface between extracellular and intracellular signaling. The signal molecules, which are called ligands in this context, attach to the receptors and thereby are localized to the cell. By analogy with the different

kinds of signal molecules, we have two fundamentally different kinds of receptors, the first kind are the *cell surface receptors*. They bind hydrophilic molecules and there are three major classes:

G-protein-coupled receptors form a family with over 100 members. All are coupled to a G-protein, which, upon binding of ligand, dissociates into subunits, which are freed and go on to activate target proteins in the cell, like enzymes or ion channels.

Catalytic receptors are either enzymes, or associated with enzymes. The group is heterogeneous, although most of them act as tyrosine protein kinases. The catalytic function is activated upon ligand binding, usually by dimerization and autophosphorylation.

Ion-channel-coupled receptors are also called transmitter-dependent ion-channels and take part in the fast synaptic signal process mediated by neurotransmitters. Upon binding of ligand, they open the ion channel.

The second kind of receptors are the *intracellular receptors*, which bind hydrophobic molecules that are able to pass the cell membrane. Some of these receptors are bound to an inhibitory protein complex, which masks the DNA binding region or prevents their translocation into the nucleus. Upon binding of ligand, they dissociate from the complex, form homo- or, more often, heterodimers, and bind to DNA. These receptors are built in a modular way, consisting of a short DNA-binding domain, a ligand binding domain and a transcription-activating domain.

1.2.5 Intracellular signaling

The means of intracellular signal processing are just as diverse as the various ways of acknowledging the signal. In many cases the signal leads to a transient modification of signaling molecules downstream in the pathway, which changes their state from inactive to active.

G-proteins exchange GDP with GTP upon stimulation by cell surface receptors. The GTP-bound form is active and dissociates from the receptor. The autocatalytic activity of the G-protein finally hydrolyzes GTP to GDP and phosphate again, turning the signal off. Active G-proteins have a range of effects, depending on the type of the G-protein. G_s for example activates adenylyl cyclase, which in turn transforms ATP to cAMP and phospholipase Cβ, which splits PIP_2 to IP_3 and diacyl glycerol.

Adapter proteins can bind to structural motifs of several other proteins, collecting them into spatial proximity. This assembly may serve to localize some proteins to certain areas in the cell or to aid in their interaction.

Second Messengers are non-protein molecules that participate in the intracellular transduction of a signal. The name stems from the extracellular signal molecule that binds to the membrane receptor and is a pathway's 'first messenger'. A major advantage of second messengers is their small size and water solubility that allows rapid diffusion throughout a cell's cytoplasm. The most important ones are IP_3, calcium and cAMP. cAMP for example activates protein kinase A.

Protein kinases are enzymes that phosphorylate target proteins. About 1% of our genes is thought to code for them and a single mammalian cell might have more than 100 different active ones. Most kinases inside the cell are serine/threonine kinases (S/T kinases). They phosphorylate proteins on their serine or, sometimes, threonine side chains. The phosphorylation changes the structure and behavior of the target. Prominent examples are protein kinase A, protein kinase Cβ and Cγ, mitogen activated kinase. Most tyrosine kinases (Y kinases) are receptors or receptor associated.

Protein phosphatases catalyze the reverse reaction of phosphorylation, the hydrolytic removal of a phosphate added to a protein. The important function of protein phosphatases is that they introduce reversibility to the protein-kinase-mediated phosphorylation of a protein, thus contributing to the dynamic nature of a cell.

Transcription factors are proteins that bind to a *cis*-regulatory element on the DNA and thereby, directly or indirectly, affect the initiation of transcription. Prominent examples are CREB, NF-κB or AP-1.

1.2.6 Nature of the cytosol

The cytosol is the matrix which occupies the space between the organelles in the cytoplasm. At one time it was thought to be a fluid unstructured matrix consisting of soluble enzymes, low molecular weight metabolites, ions and water. It is now known to have a more viscous, gel-like nature. About 20% of the cell's wet weight are protein.[Seg75] This is in sharp contrast with the highly diluted, well stirred aqueous solution that is assumed when considering classical reaction dynamics between solubilized molecules.

In addition, some molecules may only be present in a low number of copies. Both these facts make it difficult to apply the classical framework of reaction kinetics in this environment, as the basic assumptions used to derive it do not hold.

Since rate constants are usually measured in diluted aqueous solution in vitro, it is questionable how justified it is to apply them to the situation in the cell. They are used as these data are the only ones available.

1.3 Object-Oriented Databases

An object-oriented database management system (OODBMS) is a DBMS that makes use of principles derived from object-oriented programming. The first criterion translates into five features: persistence, secondary storage management, concurrency, recovery, and ad hoc query facility. The second one translates into eight features: object identity, complex objects, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility and computational completeness [ABD$^+$95].

The features typical for database management systems are:

Persistence is the ability of the programmer to have the data survive the execution of a process. In OODB, each object is allowed to become persistent independent of its type without explicit casting.

Secondary Storage Management means performance features, such as index management, data clustering, data buffering, access-path selection, and query optimization, which are hidden to the user.

Concurrency is the ability of the DBMS to offer all users working simultaneously the same level of service.

Recovery means the software is able to bring the data back to some coherent state in case of hardware or software failure.

Object-oriented features are the features found in objects of modern object-oriented programming languages. It is necessary to support these if we want to store and retrieve the language objects transparently.

Object Identity means the object-oriented model is identity-based, each object has an *object identity (OID)* which uniquely identifies this object, independent of the values stored within. The OIDs are issued and managed by the OODBMS.

Complex Objects are built from simpler ones. The use of complex objects improves the capability of representing the real world. One important feature of object-oriented databases is navigation by *reference*. This means that the objects in the database contain pointers to each other. If one object is referencing another one, we can get the second object directly by resolving the reference, instead of posing a query for its OID.

Encapsulation means that an object contains both programs and data and offers to the world an interface hiding the implementation part. The interface part is the specification of the set of operations that can be performed on the object; the implementation part describes the implementation of each operation. In most OODB, even data specification is part of the interface.

Types summarize the common features of sets of objects. A type can be considered a set of values and a set of operations on those values. This can insure type-safe programming. However, the representation of types, *classes*, can be separated from the notion of type allowing many representations per type while still maintaining reasonable type-safety. Also, one class can implement several types, as long as it satisfies the requirements defined by those types. Classes serve as templates to produce objects, an object is an *instance* of a class. For most practical purposes, types and classes are equivalent.

Inheritance is the ability of a subclass to receive all data and operations coming from its superclasses. It helps to achieve code reusability, and leads to a better-structured and more concise description of the real world and the shared specifications of applications.

Overriding, Overloading and Late Binding : When a single identifier is bound to different operation codes in different types we say that the code is overridden. When the identifier name is identical but the parameters differ the operation is overloaded. To provide this functionality, code is not bound to operation identifiers at compile time but at run time, performing the so called late binding.

Extensibility : Predefined and user-defined types must have the same status. They must be supported by the system as completely equivalent.

The standard for object databases is the ODMG Standard [Ce97]. It is a de-facto standard published by the Object Database Management Group, a consortioum of representatives for the major object-oriented database manufacturers. Adherence to the standard varies significantly between vendors. The ODMG Standard covers areas such as the Object Model and how it relates to the OMG object model, Object Specification Languages, the Object Query Language (OQL) and language bindings for C++, Smalltalk and Java.

In general, data in databases can be related by one-to-one, one-to-many and many-to-many relations. In biology, most relations tend to be of the last variety. To implement many-to-many relations between two types in an object oriented database collection attributes in one of these types are used.

Chapter 2

RESULTS

2.1 Data model for the signaling network

This section outlines the central design of the database structure, presenting in detail some of the classes that make up the structure and discussing design decisions. A full class layout and detailed descriptions of all classes can be found online under `http://transpath.gbf.de/intro/tech/api`. The database implemented with this structure has been named *Transpath*.

In this thesis, a *path* is a linear series of interactions and molecules that connects two molecules. A *pathway* is the sum of all such paths downstream or upstream of a molecule. A *network* consists of all molecules and reactions that are somehow connected. [1].

The obvious representation of a network is a graph. Compared to storing the network as a number of predefined paths or pathways, it has the advantage that there is a sound mathematical and algorithmic background that can be used for analysis.

Thus, the model is essentially a big graph. Since molecules and their reactions are the objects of interest, they will form the nodes. The resulting graph is directed and bipartite, with one node type for the molecules and one for the reactions, and edges for the connections between them. Fig. 2.1 shows an example for such a graph, that has been enhanced by family relationships.

2.2 Querying the signaling network

To make a database usable, it must be possible to retrieve the data of interest. The system must be able to answer questions the user has about the data. Elementary questions that we can ask about signal transduction data include:

- Are *items with a certain property*, or combination of properties, in the database? How many? What are these items? For example, the user could search for all molecules that match a name, or contain a certain amino-acid sequence. He could search for all molecules that have references by a favourite author, or that have a minimum number of incoming reactions.

[1] Compare [KZL99] for a similar definition

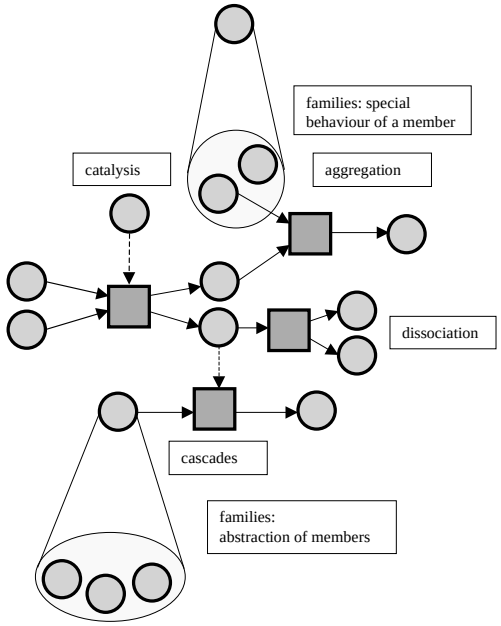

Figure 2.1: A simple network

- What are the *differences and similarities in the networks* between two different species, or cell types? For example the user could compare the known networks of mouse and human to find signal molecules that are missing in only one of them, and use that as a hint to look for the underlying genes in the other.

and

- What does the network *around, upstream or downstream* of a certain molecule look like? For example, the user could be interested in the downstream signals mediated by a certain receptor, or in the upstream pathway that regulates a transcription factor of interest. He could want to see the reaction environment around a central signaling compound like PKA, to better understand the context in which it functions.

- Is there a *path* between two molecules in the database, and if so, what does it look like? Are there alternative paths? Is there a bottleneck through which all those paths lead? What is the shortest path? For example, the user could search for paths between a certain hormone and a certain transcription factor. He could look for a molecule which occurs in all these paths to find a point where a knock-out mutation might be sucessful in breaking that connection.

- Do the paths downstream of two components *meet* at some point? For example, the user could search for such crosstalk points for certain combinations of ligands that together show a different cellular response than each one alone.

For tabulated data stored in a relational DBMS, the SQL standard [SQL97] has been established as a query tool, and is highly successful, mainly because it allows us to answer many of the questions about this kind of data that people have come up with. There is an analogue language for object oriented databases, OQL[Ce97], which implements a subset of SQL.

Other query possibilities of object-oriented databases include named objects and navigation by reachability. Named objects are nearly useless as a query mechanism: the user has to match the name exactly to get a positive result. One can obtain similar results using an OQL query on a name attribute, which in addition allows wildcard matches and use of synonyms. Querying by reference needs additional software that implements the actual query.

Of all the questions listed above, only the first two can be answered with the current crop of query languages. The last three questions need traversal of the network structure, which in turn needs an equivalent of recursion and is not in the current SQL standard. Still, they are valid and natural questions about network data, and if we want the database to be useful to the user, we have to find a way to answer them.

A metaphor that allows queries of this type is found in graph traversal algorithms. I call this a *pathway query*, to discriminate it from the classic OQL-like queries. The depth-first and breadth-first search algorithms[Knu97, Sed92] create spanning trees through the network graph. Since our graph is directed, it is possible to restrict them to process along only one direction. This approach directly allows us to answer the first of the three questions.

Once we accept this metaphor, it is easy to extend it to allow more elaborate and complicated kinds of queries by selecting certain nodes in the search. Two central extensions are *filters* and *boundaries*. Filters act to highlight or select certain nodes or paths that have been observed in the traversal, while boundaries limit the search to a subgraph.

There are numerous criteria to decide which nodes will be chosen for filtering or bounding. A general one is the definition of sets of molecules and reactions by classic queries or the results from another pathway query. Another possibility comes from the topology of the network: nodes that are within a certain distance from a starting node, nodes that have been observed before in the search and therefore constitute an alternative route or feedback, or nodes that have a certain number of inputs or outputs can be selected.

When we combine the basic traversal algorithms with these selection approaches, it becomes possible to answer the last two of the three questions, too:

For example, in the second of the three questions it is possible to first select a transcription factor from the database, and then use it to filter all pathways that extend downstream from some extracellular hormone, yielding only those paths that lead from the hormone to the factor. In the last of the three questions, one can use the downstream cascade from one component as a boundary for a downstream cascade from the other component, and select all molecules found on that boundary, yielding the crosstalking molecules.

To take into account family and state relationships in the database, we have to extend the search to traverse these relations, too. Again, there should be a possibility to limit the depth – or height – of these classification transitions.

Section 2.5.3 contains a review of the implementation of these concepts.

2.3 Visualising the signaling network

Once we have a means to pose pathway queries, we have solved half the problem of making the data accessible. It is as important to have a metaphor for visualizing the database contents. There are different ways to do this, and they vary in terms of their ability to depict the network.

We provide five different views on the data, in order of increasing visualized context:

1. tabular data that present the attributes of a single object in the database, without further network context. (Fig. 2.12).
2. Linear paths that show an ordered list of interactions connecting two molecules of interest. (Fig. 2.19).
3. Tree-like cascades that show which elements are downstream or upstream of a core component. (Fig. 2.14).
4. Graph layouts that show the full network as connections between nodes, according to its topology. (Fig. 2.17).
5. Hand drawn, clickable maps that enrich the topological layout with additional information, like subcellular location or molecule type. (Fig. 2.11).

2.4 Class layout for the Transpath database

The aim of the class layout for the database is to capture the structure of the biological relations that are relevant for signal transduction. This section will present my insights on the correct class hierarchy to capture signal transduction pathways. The differences between the current prototype implementation and this layout are discussed in chapter 4.

Suppose we represent molecules as instances of a Molecule class, and reactions between them as instances of a Reaction class. If we wish to represent a many-to-many relation between molecules and reactions, it is sufficient to provide in the Molecule class a collection attribute that can hold reactions [Heu92]. Since any molecule will have its own collection, we have achieved a many-to-many relationship, where each molecule can refer to many reactions, and many molecules can refer to each reaction.

Note that this relationship is directional. Starting from a given molecule, we find the set of related reactions easily, because molecules are pointing to reactions. Starting from a given reaction we cannot easily find the set of related molecules, because reactions are not pointing back to molecules. To find the set of molecules, we must examine all molecules and remember the one that pointed to the reaction in question, which is similar to the way queries in relational databases are used to pull together related bits of information.

One of the fundamental ideas in databases is to remove redundancy to avoid contradiction or inconsistency. If there are two redundant items and we accidentally change one of them, they will afterwards contradict each other, and the database will be inconsistent.

To enable faster navigation by reachability in any direction, especially for the creation of pathways, we introduce an additional, redundant collection of

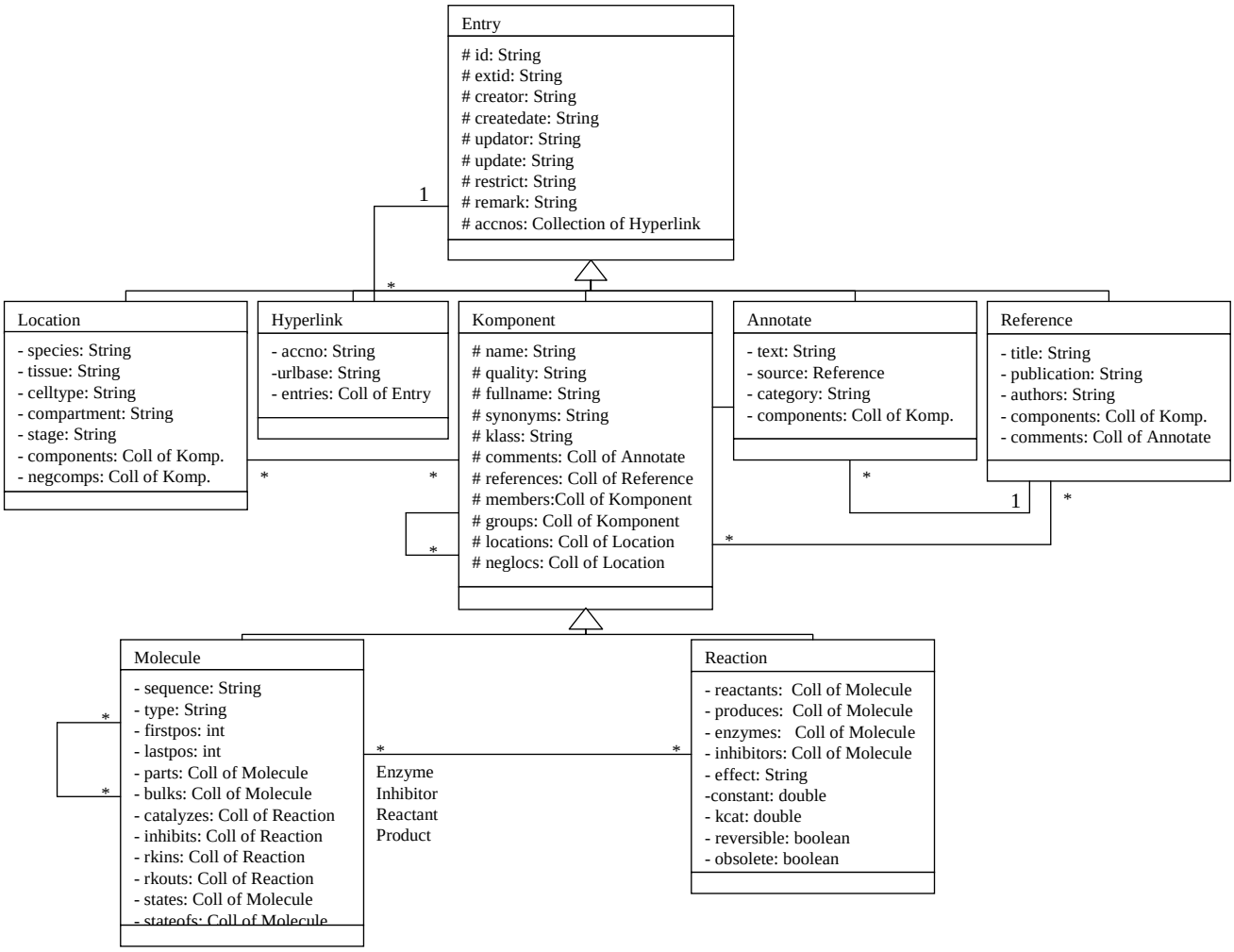

Figure 2.2: UML class layout of the current implementation. Only one relation is shown between two classes, even if several exist. For the key relations between Molecule and Reaction the roles are given. To maintain clarity, methods for the classes are not shown.

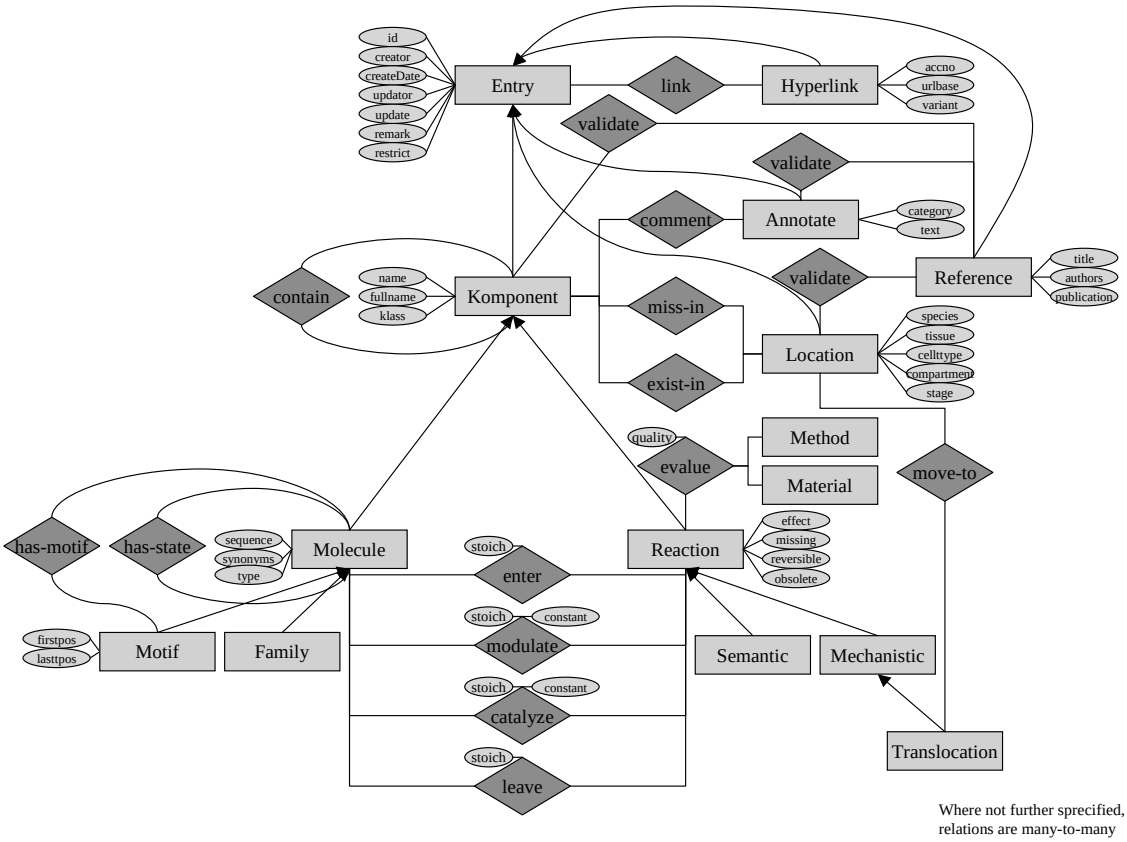

Figure 2.3: Entity-relationship diagram for the Transpath database.

backreferences for each collection of references. This controlled redundancy improves the database performance. To guard against consistency errors, we use encapsulation: an object cannot directly be stored in or removed from these collections, only through methods which automatically insert or remove the cross-references on that object.

Cross-referencing is so commonplace for object-oriented databases, that it is supported by the ODL (Object Definition Language) recommended by the ODMG.[Ce97, KE99]

It is not possible to associate additional information with this kind of connection between two items. Since the pointers that refer to the items are internal pointers of the programming language or OODBMS, we cannot store data in them. This is in contrast with relational databases, where it is easily possible to extend a linking table with additional fields to store contextual information. If we want to mimic this possibility with an object-oriented database, we have to introduce an additional class that emulates this linking table. We do this in the signaling class hierarchy for the central relation between Molecule and Reaction, which needs additional properties to adequately represent the biology (see 2.4.8 below).

2.4.1 Entry

All classes in the database inherit from the superclass Entry. It has no biological meaning, but allows us to reuse the fields that hold administrative information on each entry, like who created it and when it was last modified. There are no direct instances of this class in the database.

2.4.2 Component

The class Component represents nodes in the graph. It is the superclass of Reaction and Molecule, and allows the reuse of fields that apply to both kinds of nodes, like name and family membership. Component has no direct instances in the database.[2]

Groups

Important attributes of all components are the collections that allow expression of family relationships. Each Component can act as a group of other components or as a member in such a group. In the case of Molecule, this grouping represents family relationships between the molecules. In the case of Reaction, this grouping can be used to assign reactions to a general class of reactions, as is done successfully in the EC nomenclature [otIUoBB92]. The third possibility is the use of mixed groups, which contain both reactions and molecules, and correspond to nets or explicitly stated pathways. This is interesting since it would support the appealing idea of building networks from subnets in a modular fashion [MA97].

Locations

Each component in the network has two lists of Location objects (see 2.4.9). One list contains all locations where experiments have affirmed the presence of the component, the other list contains all locations where experiments have affirmed its absence — which is neccesary to distinguish between cases where the component is absent and cases where its status has not been investigated. For molecules, the locations describe where the molecules have been shown to be expressed, or have been shown to be not expressed. For reactions, the locations describe in what kind of system the reactions have been confirmed, or have been found not to happen.

2.4.3 Molecule

Molecules interact with each other to build pathways. A Molecule is anything that is subject to reactions. Most Molecules have a mass, be it a small molecule like ATP, a protein or a stretch of DNA sequence. No difference is made between receptors, enzymes, second messengers, transcription factors or other special kinds of proteins. A Molecule can also be a group of such entities, like a protein family, a state of such an entity, like the phosphorylated form or a complex of several other Molecules. And finally a Molecule can be part of another Molecule, either non-covalently bound as in a complex, or covalently bound as

[2] Component was named with a K in the Java implementation to avoid confusion with the Java Component class.

in a structural motif of a protein. The reason for such a wide scope for this class is to catch anything that shows specific signaling behavior.

A molecule can serve in four fundamental roles in the context of a reaction: it can be an educt, a product, an enzyme or a modulator. Enzyme, educt and modulator are the inputs of the reactions, while products are the outputs. For semantic reactions (see 2.4.7) molecules are only grouped into signal donors, which are the inputs, and signal acceptors, which are the outputs.

We use two other collections, states and stateofs, to keep the states (see 2.4.6) separated from the family grouping hierarchy (see 2.4.5).

2.4.4 Motif

Motifs of a protein often are responsible for its signaling behavior (see [Hun00]). A single protein can have several signaling motifs, and some motifs are shared by several proteins. Sometimes, a motif is known to be responsible for signaling, and it should be possible to link the signaling reaction to the motifs instead of the molecule. This makes it simpler to represent molecules that have several, independent motifs.

Since there is a many-to-many relationship between motifs and proteins, we add a collection of motifs to Molecule. A motif is represented by a Motif entry, which is a subclass of Molecule to make it possible to use it in signaling cascades.

2.4.5 Family

Grouping molecules and motifs into *families* is essential for a usable signal transduction database. When the family relationships are stated in a formal way, it is easy to write algorithms that exploit them in user queries, see 2.5.4, while marking inherited properties as inferred information.

Family specifically addresses the issues of

1. molecule classification
2. paralogues[3]
3. orthologues

The family relation is implemented by the groups/members collections mentioned in 2.4.2. In the molecule classification, the last classes before the individual molecules are groups of paralogues. The same classification is used to handle the orthologues, grouping them under a species independent group entry. By overloading this collection we avoid creating yet another hierarchy and can extend the mechanisms for paralogue handling to orthologues.

Paralogues

Due to gene duplication and mutation within species multiple paralogues have often evolved. For a single gene, there may in turn exist different splice variants. Sometimes in the literature a signaling activity is first attributed to a

[3] Orthologue and paralogue: A pair of genes are orthologous when they are different because of a speciation event, and are paralogous when the difference resulted from a gene duplication event.

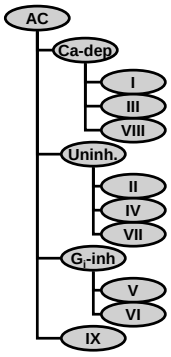

Figure 2.4: Adenylate cyclase family tree. Ca-dep. are the calcium dependent forms, Uninh. are the forms without inhibition mechanism and G_i-inh. are the forms inhibited by $G_{i\alpha}$ or calcium.

single molecule, and later it is discovered that there is a whole group of similar molecules.

For example look at adenylate cyclase (AC, Fig. 2.4). There are at least nine isoforms, ACI–IX), which are all activated by $G_{s\alpha}$, while only isoforms I, III and VIII are Ca/CaM dependent, II, IV and VII have no inhibition mechanism and V and VI are inhibited by $G_{i\alpha}$ or calcium.[SDG96]

We represent this kind of hierarchy in the database.

Orthologues

In many cases, reactions between proteins have only been shown for one of several model species, like rat, clawed frog or fruit fly. It is common to scan sequence databases for orthologue sequences and assume, if these can be found, that they play a similar role in signal transduction. This orthologous grouping makes it possible to stitch together pathways from bits and pieces that have been investigated in different species. We pragmatically add orthologues to the family tree for paralogues, by grouping them under one of the entries in the tree, usually at the isoform level.

Classification methods

Since it is easier to examine the sequence of a gene or protein than to investigate its function, a certain behavior is usually shown in detail for only few members of a protein family, and homologues are added to the functional group by sequence or structural similarity.

Based on this assumption, various good databases exist that try to classify proteins and map sequence motifs to functional annotation [BBD+00, CGK99, HHP99, HBFB99, Apw00]. They cluster proteins by multiple sequence alignments and use common structural motifs, "profile" patterns, or Hidden Markov Models derived from these alignments to classify new proteins. Sometimes these methods can predict the function correctly, but sometimes not.[4] Thus it is com-

[4]If the behavior would always be predicted correctly, the automated construction of a signal transduction database would be feasible: a whole signaling network content could

mon practice to group molecules into families on the basis of sequence similarities, even if they do not share common behavior.

For a signaling database, it is advantageous to group molecules that show common signaling behavior, that is, to group them by function. Since it is the function we are interested in, we would like to group only by function. On the other hand, we would like to draw as much as possible upon expert knowledge, and stay coherent with the groupings that already exist. To solve this dilemma, the best option seems to be that we group the molecules as is done traditionally, but link signaling only to those molecules for which it has been shown.

Given the tree for adenylate cyclase in Fig. 2.4, if the calmodulin-dependent reaction has been demonstrated for ACI, we link that reaction to ACI only. We link statements made on a generalized level, like the ones from review papers, to nodes on a higher level in the molecule hierarchy. Given the tree, we can for example link the general $G_{s\alpha}$ activation to the AC group. The context of the original literature statement is preserved.

2.4.6 State

The unmodified form of a protein and all its modified forms are its *states*, where the modification can be by covalent binding, by complexation, or by change of environment. The protein *per se* is a concept that is based on the observation that there is only one *gene* coding for each protein sequence. All the states share the same gene and consequently part of their structure, the amino acid chain. They are functionally related, often even reversibly transformable into each other.

A *basic* Molecule entry captures this concept and is the class of all states of a protein. These states are different molecules, and we store them as different Molecule entries. As Molecules, they can be used in a pathway assembly. We store general information like the amino acid sequence in the basic Molecule entry and link its states to it.

In the simplest case there are only two states, an inactive one, and an active one. In other cases, there are more. For example, a transcription factor can be

1. de-phosphorylated in the cytosol
2. phosphorylated in the cytosol or
3. phosphorylated and bound to DNA in the nucleus

to name a few possibilities. The same protein will exhibit distinctly different signaling functions in these three states. For example in state 1 it will be susceptible to phosphorylation, in state 2 to translocation into the nucleus or dephosphorylation and in state 3 it will activate transcription.

The number of states for a molecule is the product of the number of its modified forms and the number of locations (see 2.4.9) where it is found. Only compounds which share the same location interact in nature.

It is impractical to enter a separate state for each location. Most molecules can be found in several tissues, at several development stages, in several cellular compartments, several organs and several celltypes. To enter a state Molecule

automatically be generated from the sequence databases.

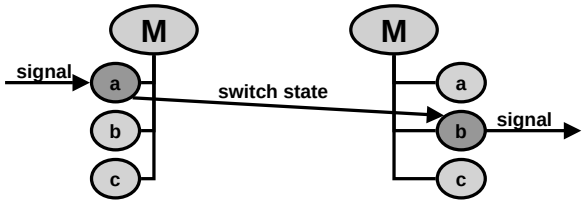

Figure 2.5: State switching

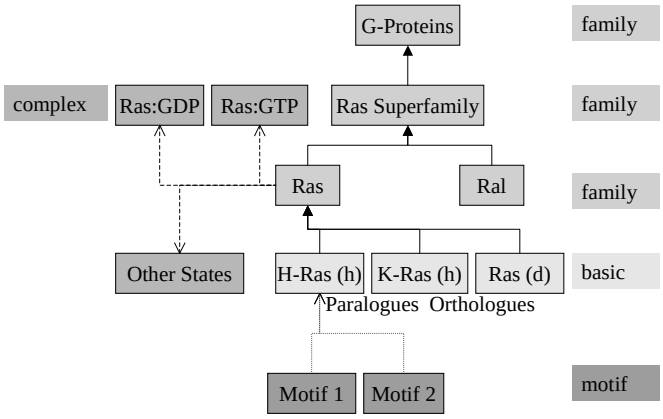

Figure 2.6: Hierarchical relations and roles for Molecules, shown for a subset of the Ras superfamily. Basic Molecules are translated proteins or small molecules that have mass. Family Molecules are groups of related molecules or of other molecule groups. Motifs are structural or sequence motifs of basic Molecules. Complexes and other states change the availabiliy of the molecule for reactions.

for each possible combination would lead to an explosion in the number of states, and redundancy in the reactions. This problem is circumvented by using the location list inherited from Component.

In each state, the molecule is available only for a subset of all reactions for that molecule. Receiving a signal changes the molecule's state, usually leading to a new state from which reactions are triggered (Fig. 2.5).

When the molecules assembled in a complex show different signaling behavior from the independent molecules, this complex is a state for them. We link it as such to all its constituent member molecules.

Fig. 2.6 sums up all the various hierarchical relations that Molecules in the database can have.

2.4.7 Reaction

Reactions as processes are not physical entities like molecules, yet they are the central point in a signal transduction database. By representing these reactions between molecules as separate nodes in the graph, it becomes possible to store their properties and annotate them.

Since many reactions in signal transduction are catalyzed, and most catalyzed reactions are quasi-unidirectional, all reactions stored in the database

Semantic reaction	Mechanistic Reaction
Signal flow	Mass flux
Implied states "active" and "inactive" for the molecules. The Reaction is linked to basic molecule entries.	States are explicitly stored in the database. The Reaction is linked to state molecule entries.
Emphasise meaning of reaction for molecules.	Emphasize meaning of molecules for reaction.
Only two roles for molecules: signal donors and signal acceptors.	Four roles for molecules: reactants, enzymes, modifiers, and products.
Easy to read or enter, less detail.	Hard to read or enter, more detail.
Useful for extracting review literature and proven influences without known mechanism.	Useful for extracting primary literature and interactions with information about mechanism.

Table 2.1: Semantic and mechanistic reactions in comparison.

are unidirectional. Equilibrium reactions are stored as two reactions, one for each direction.

There are a semantic and a mechanistic view to reactions in the database. Both views of the Reaction have their merits and are used. Table 2.1 summarizes the properties of the two views, which are discussed in detail in the next sections, and Fig. 2.7 shows the difference graphically.

The basic molecule entry should be used in the semantic reactions. The states should be used in mechanistic reactions. To avoid a mixing up of concepts, the basic entry is not used as any of the states. In particular, the basic entry is not used in place of the unmodified form.

Semantic reactions

The representation of reactions commonly encountered in scientific review papers can be seen in Fig. 2.8. I call this representation *semantic*, because it assigns a meaning to the states of the molecules for the overall network, "active" or "inactive". It is easy to understand and it is familiar to scientists from the papers. It shows how the signal flows through the network. The reactions need additional information, namely if they activate or inhibit the target molecule.

The reactions in this scheme are binary. Molecules act as signal donors or signal acceptors to the reactions.

This view implicitly assumes that each molecule exists in an active and an inactive state. It is not necessary to differentiate between them, as it is understood that incoming activating reactions always refer to the inactive state, while incoming inactivating reactions and outgoing reactions refer to the active state. Because the states are implied for semantic reactions, we link them to basic Molecule entries in the database.

Saying that a molecule has an "active" or "inactive" state is a semantic statement. Both states undergo reactions, and the decision which is which can consequently only be determined in the larger context of the whole network.

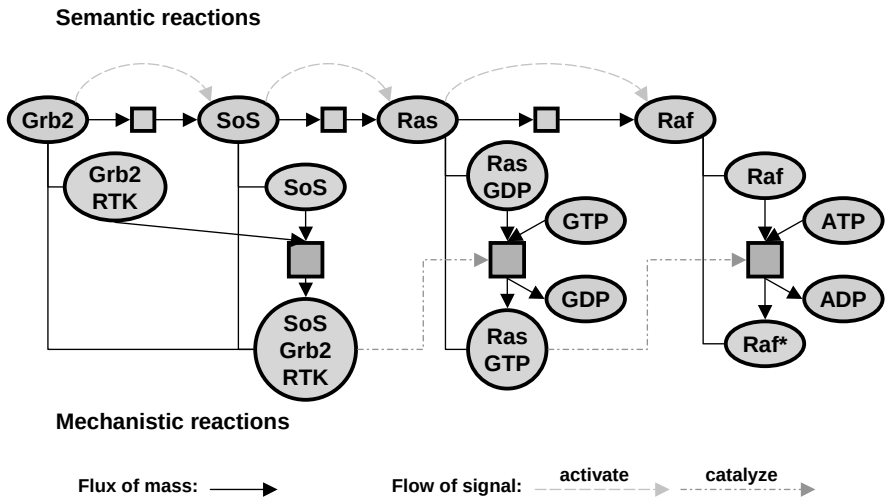

Figure 2.7: Semantic and mechanistic reactions. Semantic reactions (small, light squares) go from Ras to Raf and from Raf to MEK, implying that each of those molecules has two states, one "inactive" for receiving signals, and one "active" for passing them on. The other Reactions (darker, bigger squares) are mechanistic and assign states explicitly to the molecules.

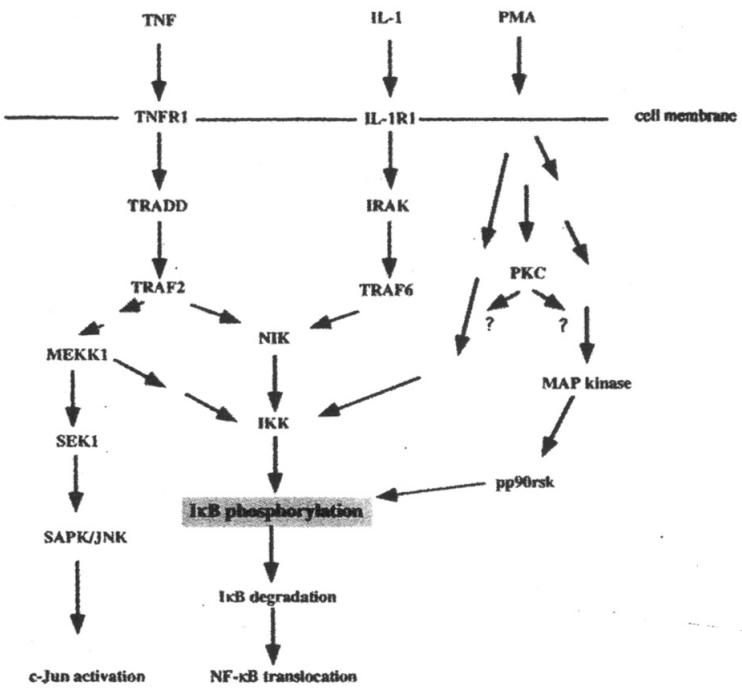

Figure 2.8: Typical data from review literature. Putative pathways to NF-κB activation [SB97]. Arrows are activating reactions. Multiple arrows indicate multiple steps, question marks uncertainty.

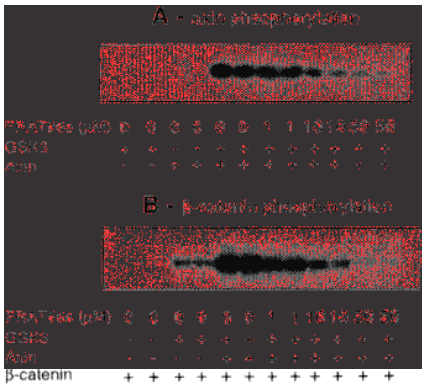

Figure 2.9: Typical data from the primary literature. The data show that β-catenin and axin are phosphorylated by GSK3, and presence of axin increases the phosphorylation of β-catenin. FRATtide inhibits the phosphorylation by competition [T$^+$99]. Another paper [ABS$^+$97] then shows that the phosphorylated form of β-catenin is ubiquitinated and degraded by the proteasome.

Mechanistic reactions

The classical chemical reaction notation

$$\nu_1 X_1 + \nu_2 X_2 + \ldots + \nu_n X_n \overset{E}{\rightleftharpoons} \nu'_1 X_1 + \nu'_2 X_2 + \ldots + \nu'_n X_n$$

can model aggregation into a complex, dissociation of a complex, chemical modification and catalysis in the detail necessary to capture data from the primary literature. An example for this kind of data is given in Fig. 2.9. I call this representation *mechanistic*, because it reflects how the molecules interact, and not how the signal is transported. It is well-established and familiar.

Molecules act as enzymes, substrates, modulators or products, and any reaction has exactly one or no enzyme, one or more substrates, any number of modulators, either inhibitors or activators, and one or more products. There is semantic information in assigning some of the inputs to the enzyme or modulator collections, which tells us something about the local meaning of the molecule for the reaction.

This view is more tiring to read and enter, and demands a more detailed knowledge of the mechanisms of transduction, because it shows in more detail what happens. It makes no assumptions about the molecule states. All the different states of a molecule have to be listed as separate entities, and in the database we link the mechanistic reactions to state Molecule entries.

Translocations

In a *translocation*, the same molecule enters and leaves the reaction with a changed spatial context. This process takes some time, which is important for the dynamic behavior of the network.

Translocations cannot be represented in the basic mechanistic model which assumes all reaction partners are present in the same reaction space. Since

a molecule is just associated with a list of locations, we cannot differentiate between, for example, the cytosolic and the nuclear form of the molecule. A reaction that moves a molecule from one form to the other is then a loopback from the molecule to itself. Therefore we either have to associate the two locations with the Connection entries between the Molecule and the Reaction (see 2.4.8), or subclass Reaction with a class Translocation, which provides two lists of locations, one for the entering and one for the leaving molecule. The latter solution is chosen in Transpath.

2.4.8 Connection

Connections are binary relations. They represent directed edges in the graph, where one end node is always a molecule, the other a reaction. Thus there are two kinds of Connections, one per direction.

Connections are important because they can contain additional information that is dependent on the context between a molecule and a reaction, like stoichiometric factors or location information. Connections can store kinetic parameters for reaction modifiers, for example the K_i constant of a competitive inhibitor, which is dependent on the inhibitor and the reaction.

Connections can help to depict translocation reactions, by associating one location with the incoming and another location with the outgoing connection of a reaction.

2.4.9 Location

Location describes spatial and temporal environments. It tells apart molecules and reactions found in different kinds of cells, tissues and organs. Without it we would get the model of a super-cell, which can do everything. Imagine the network for each cell being drawn on an overhead-sheet. Then the view without location information would be all these sheets layered on top of each other.

Signaling needs the location context. Subcellular location plays an important role in the activity of many compounds. For example transcription factors like NF-κB are only active in the nucleus, Ras is only active after recruitment to the inner plasma membrane. Since all signaling takes place in the complex matrix that is the multicellular organism, we first need a good model of the spatio-temporal segmentation of the organism.

Relevant aspects of location include:

1. subcellular location. Examples: nucleus, plasma-membrane associated or spanning, mitochondrial, cytosol.

2. tissue/cellular location. Examples: liver, kidney, body, glomeruli, Schwann cells, adipose tissue

3. developmental stage. Example: Carnegie stages in human.

4. species. Example: vertebrata, mammalia, human, rat, mouse

The location items often relate to each other in tree-like schemes. A human for example is a vertebrate in the taxonomic dimension. Glomeruli are a substructure of the kidney. This classification is necessary to avoid state explosion

— otherwise we would have to insert a separate entry for each subtype, when the supertype is mentioned in the literature.

Sequences and molecular weights differ between orthologues. While the species is categorized as part of the location of a molecule, these orthologues are not different states of the same molecule, they are different members of a group of molecules. The molecules from different species are stored as members of their orthologue group (see 2.4.5).

The experimental proof for an Reaction is usually obtained in artificial systems. We cannot claim that a reaction takes place in a cell in vivo, simply because all involved molecules are found in this cell. We have to link the reaction to the location for which it was proven.

When we have further information that depends on the location, we store this information in the relation between Location and Component.

2.4.10 Hyperlink, Annotate, Reference

These classes are not important to model the biology itself, but they serve to integrate annotation and other resources with the pathway graph. In short Reference refers to papers from literature from which the information we present in the database was extracted. Hyperlinks connect to external databases. Annotates are user visible comments of all kinds.

2.4.11 Method, Material

Method and Material are used to assign a quality or level of trust to the experimental findings. Method lists experimental methods and Material lists biological materials used in conducting the experiments. This information is not extracted from the literature. It is provided as an additional service to the user.

The quality level can be used by the pathway builder for automated highlighting or to filter out low-trust data.

2.4.12 Signal

A signal according to Webster is:

> (...) *a*: an object used to transmit or convey information (...) *c*: a detectable physical quantity or impulse (as a voltage, current, or magnetic field strength) by which messages or information can be transmitted

In signal transduction a diverse set of molecules interacts with each other. The signal cannot be a physical part of any of these, such as a phosphate group, since these items stay localized with their carriers, while the signal travels on. So definition *a* is not useful.

Using definition *c*, the only physical[5] quantity that all the signaling molecules share and that changes as the signal proceeds is their concentration. It would be chemically correct to speak of their activity. The activity inside the cell is hard to determine, so for our purposes concentration has to be good enough.

[5] Actually chemical.

If we express the signal as the change in concentration of activated target divided by the change in concentration of signal source [KHWB97] we can quantify the signal transfer. Shannon entropy, a concept of information theory, gives the classical way to measure signal transfer, but is not so easily applicable here.[SW63]

Input signal and reaction strength determine the output signal. We find analogues for them in signaling molecule concentrations and rate constants.

Signals travel through the network. They are not static and for this reason are not stored in the database. Consequently Signal is not implemented as a class.

2.4.13 Pathway

Pathways are not explicitly stored in the database. They can be dynamically created from the underlying graph upon request. The graph incorporates all the available and atomically updatable data and allows to please differing opinions on where pathways should be bounded.

Pathways in the current implementation can be generated with the pathway builder, see 2.5.4.

2.5 Interface for the Transpath database

The interface for the database is implemented as a web interface for access over the World Wide Web. It is written using the Java programming language.

The web interface uses a 3-tiered architecture, with the computational logic in the middle tier on the server side. Fig. 2.10 shows the general layout. Server side logic has the advantage that it has direct access to the data in the database, which is faster then sending requests over the net.

Requests are sent by the browser and accepted by the web server, which channels them to one of the servlets. The servlets themselves are wrappers that either translate the parameters of the request into calls to internal APIs or handle simple requests themselves. All servlets that need to access the database are isolated from the actual database vendor implementation by the Clerk class. The Clerk finally forwards requests to the database server. The database server returns the data in the form of objects which are formatted into HTML pages by the servlets and returned through the web server to the browser.

Some servlets, like the TemplateServlet or the SetServlet do not need to access the database for their work. TemplateServlet is used for the generation of a few templates like HTML pages and SetServlet is responsible for managing transient result set data.

2.5.1 Sessions

HTTP is a stateless protocol yet there is information that we have to store about a user's session:

Several databases are offered through the same interface. We have to remember with which of those any one user is accessing.

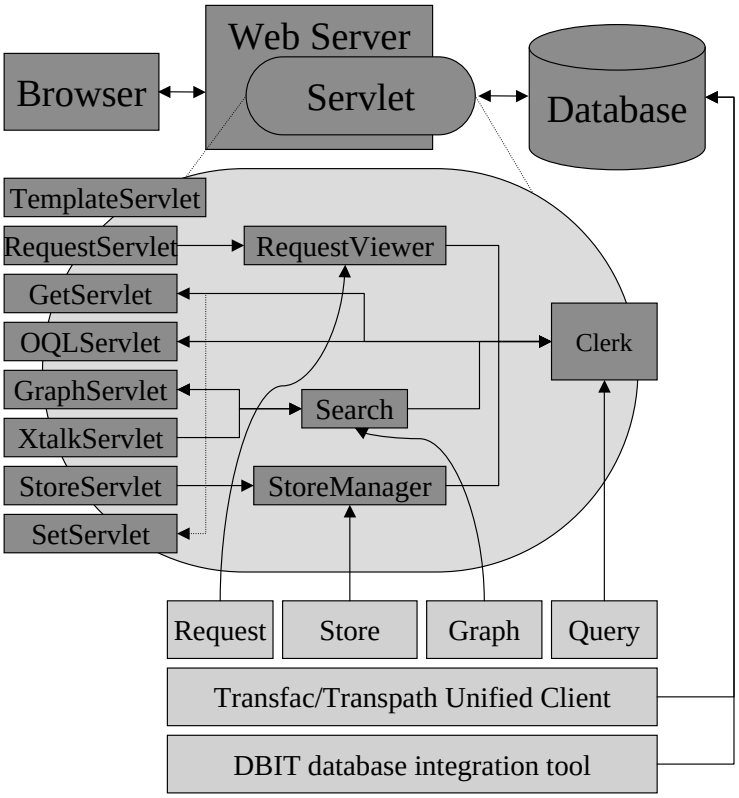

Figure 2.10: Schematic interface for the database. The light boxes at the bottom are command-line tools for the same tasks that the web interface servlets provide, a stand-alone input client application[Gro00] and an export tool to relational databases[Chr00].

Entering new data (see 2.5.2 below) is a privileged activity and needs forms instead of the normal view which merely displays the data. We have to remember which mode the user is in.

Classified information may be accessible only for certain users. We let users log in and have to remember their authorization level.

Result sets generated by requests are open for subqueries and usable as modules for pathway queries. We have to remember and manage the result sets for any one user.

Other formats for output that the user has elected have to be remembered.

To address these issues, the web interface remembers the state of a connection by a browser. It does so by using a session id, which is issued by the servlet upon the first request and sent by the client with subsequent requests. This allows storage of information about the connection on the server under this id, and saves sending it back to the client with every response, which would be prohibitive for things like large result sets.

Since not all browsers support cookies, the session id is appended to all URLs as a parameter. Enforcement of cookies would be an easier solution, less susceptible to the user using his BACK button until reaching a page where he had no session id and thereby losing his session context.

All Servlets have to be programmed in a thread-safe way, once internal state is retained. Otherwise, if a servlet is working on the request of a user and the same servlet instance is called by a second user who has chosen different parameters, the servlet's internal state will change also in respect to the first user, and the results are out of control. To avoid doing the locking and synchronisation for shared internal state variables explicitly we are careful to not store state in the servlets themselves, only in the session context.

2.5.2 Data entry with a form client

Data entry to the Transpath database is done based on XML flatfiles, which are parsed into objects and submitted to the database. This approach allows low-tech support for external updaters, which can use a local application or a plain text editor to create the files. The files then can be sent in via e-mail, and parsed into the database. The SWISS-PROT database for proteins [BA00] has operated on a similar model for several years.

For our database this approach is difficult. The data is highly interconnected, and practical experience has shown that annotators need to query and visualize the incremental updates they make while building the network. A dedicated, web-independent input client application, which also interfaces to the TRANSFAC database [W$^+$00] was developed independently of this work to integrate the two resources [Gro00]. Due to technical difficulties, this tool unfortunately proved too slow to allow sensible work.

The input client that is used for daily data input is based on the web interface and automatically generated forms. The forms are produced dynamically and reflect the fields that are available in the entry type. When adding new fields, nothing has to be changed in the input client code as it adapts automatically. Upon submission the forms are transformed into XML text and then parsed into the database by the StoreServlet.

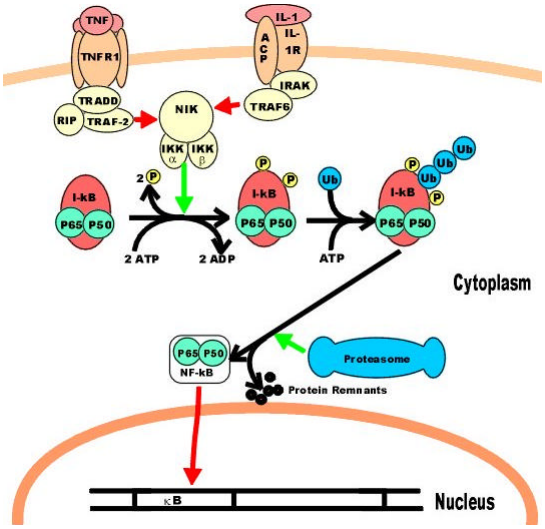

Figure 2.11: An example clickable map. This is an especially good example, since it shows not only the advantage of maps for visualisation, in being optically pleasing, but also the disadvantage, because this map is missing MyD88 at the IL-1R complex, which was more recently discovered and not entered when the map was created.

2.5.3 Web navigation interface

The user has the top level options to

1. Browse the database by using preconstructed clickable pathway maps, like the one in Fig. 2.11.
2. Query the database in the classic, text-match based way, either with a simplified query form, or with a full OQL query.
3. Pose pathway and crosstalk queries about entries identified by accession number.

Most users start by browsing the maps and clicking on an item or by looking for a Molecule of interest and choosing from the resulting list. In both cases, they end up with a single molecule entry, like the one in Fig. 2.12. From there they can either follow the Hyperlinks to connected entries, or enter pathway queries.

Users interested in automated analysis of the data can choose other formats than HTML, which are easier to parse, like a simple text flatfile format or XML. They can download the whole contents of the latest database release in XML format, if they desire to do so.

2.5.4 Graphical Pathway Builder

The pathway builder executes pathway queries and outputs the results. The default visualisation is a pathway cascade, and a second option is an automated layout by graph drawing tools.

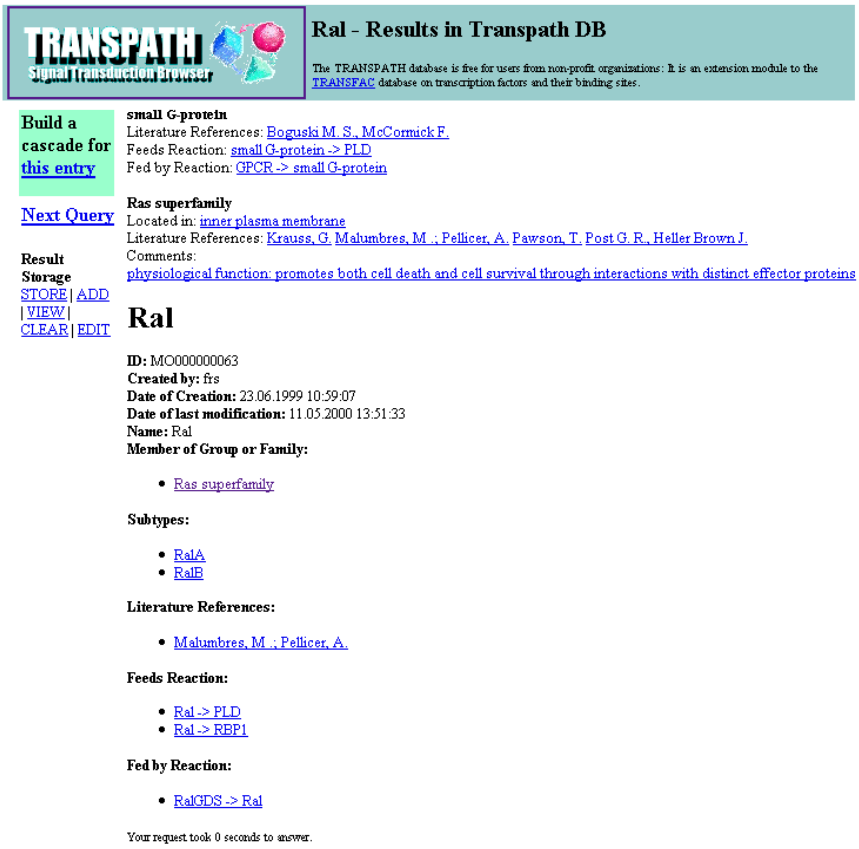

Figure 2.12: Example output for a single molecule entry. In this example, the option to show family information was selected, printing above it abstracts of families to which this molecule belongs.

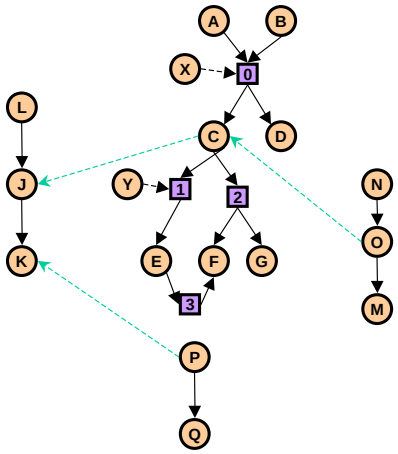

Figure 2.13: An example network. Squares are reactions, circles are molecules, dark dashed arrows are catalytic influences, solid arrows are mass fluxes and light dashed arrows are family relationships, where the superfamily is at the end which has the arrowhead. In graph terminology, the arrows are called *directed edges*, and the squares and circles are called *nodes*. The same notation will be used in the following figures.

The cascades produced by the builder are mistaken by some users as predictions of signaling behavior, but this is not the case. Pathway queries merely return subsets of stored data and then visualize them in a sensible manner. These visualisations do not constitute actual predictions of what happens in the cell, and therefore can show reaction sequences which will not normally happen. Simulatory predictions, which are covered in 3.2, need the assignment of dynamically changing activities to the molecule nodes.

Fig. 2.13 shows an example network used here to explain the behavior of the pathway builder in detail. The examples were generated with a small test database containing this network. The same procedures are of course applicable to the larger database which contains the actual biological knowledge. The figure shows that the pathway builder treats reactants, enzymes and modifiers as inputs and products as outputs. To find out the exact role of an input, the user currently has to look at the full reaction entry.

Note that the way the reactions and molecules are depicted to be wired here is a reaction-centric view. Reactants and enzymes lead into a reaction, while products are emitted from the reaction. In the database there exists an additional edge for each edge in the figure, which goes in the exactly opposing direction. If we say that we follow an edge against its direction, then we actually follow that backlinking edge. This technical detail is not needed to understand the behavior and is mentioned here only for correctness.

So, let us forget about inheritance and protein families. The central part of the image is on one hierarchy level, and staying on this level we can set a *direction* and a *distance* for the search. If the direction is

downstream, the builder follows the edges *along* the direction in which they are pointing.

upstream, the builder follows the edges *against* the direction in which they are pointing.

up- and downstream, the builder follows the edges of the starting node in both directions, so that both these trees are constructed.

reachable, the builder follows the edges in both directions *at any node*, so that the whole connected net around the starting node is traversed.

Fig. 2.14 shows the traversal in up and down direction. In downstream direction with respect to node C, node F can be reached by two paths, and the search skips it the second time around. This is necessary, since otherwise feedback cycles – imagine reaction 3 pointing back to C instead of F – would make the search run forever. That the node F has been observed before is shown in the output by setting its name in brackets. The bracketed hyperlink does not point into the database, but points to the point where the node, and consequently possible subtrees, appeared first. This makes it easier to follow the flow of the cascade. All other names are hyperlinked to retrieve the respective single entries from the database.

An important point here is that nodes D and Y are not seen in the upstream and downstream traversal, even though they are part of traversed reactions and are thereby connected to the starting node. The edges connecting them do not have the right direction to be considered by the search. This again shows that the output cascade is a way to show data, not to predict what happens biologically: reaction 1 would probably not proceed in the absence of its enzyme.

The right side of Fig. 2.14 shows the concept of search distance. The search distance is the number of edges that will be traversed away from the source node, before the search skips further subtrees. Since node F would be at a distance of six following the path from A over C and E, in the example with distance set to four, the search stops following this path before reaching node F. On the path over C to F directly, which is the shortest path but is traversed later in the normal search[6] F is within the distance of four and is shown in the output.

The search distance limit is useful, as the output tends to get too long to be useful without it. If the user wishes he can switch it off by setting it to “any”.

Fig. 2.15 shows the traversal with the “reachable” option. The output uses another symbol for the reactions to make clear that no direction is shown. If the distance were greater than two, we would observe a lot of bracketed instances of the three already traversed reactions in the output, since they can be reached again from the current leaf molecules, by following the edge that led to those molecules in the backward direction.

Searching for the whole connected net generates unwieldy outputs even for small distance limits. It still is a useful option, if the aim is to extract the whole subgraph that is connected to a node, for example for further analysis, or visualisation by other tools.

[6]The order of traversal in the depth-first algorithm employed for the normal search depends on the order in which neighboring nodes for the current node are listed. Since we do not guarantee any order for the collections in our class definition, we can not rely on having a certain order. That the shortest path is found later here is coincidental, but the larger the number of possible paths gets, the more likely we are to traverse a longer path first.

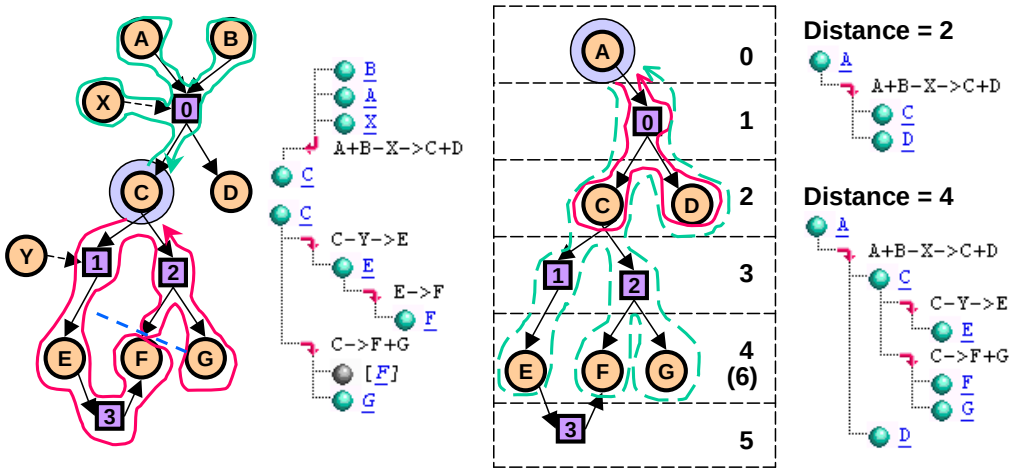

Figure 2.14: Search direction and distance. Notation is the same as in Fig. 2.13. *Left side*: starting from node C, the upper half shows the traversal in upstream direction, the lower half in downstream direction. Next to the network the output produced by the pathway builder is shown. In this output, underlined text provides hyperlinks for molecules. Clicking it retrieves the Molecule record from the database. Bracketed, italicized text provides hyperlinks for molecules that have been seen before in the output. Clicking it jumps to the line where the molecule was encountered first, is hyperlinked to the database entry and may have further sub-pathways. Dashed line indicates limiting the search because of a previously seen node. *Right side*: starting from node A, a traversal with distance 2 is shown as a solid line, another one with distance 4 is shown with a dashed line. The numbers in the dashed boxes are the distances of the nodes in the box from node A.

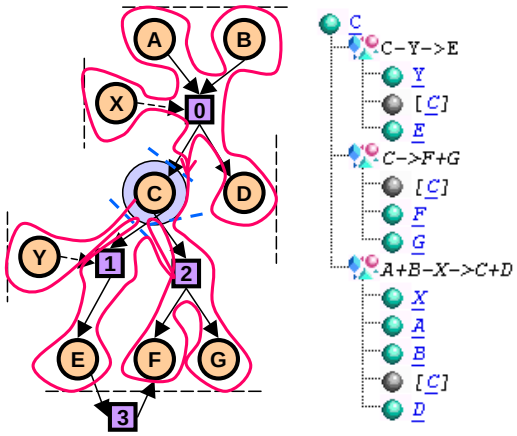

Figure 2.15: The subnet for the *reachable* direction with a maximum distance of two. Thick light dashed lines show where the search skipped because the node was seen before (around the starting node, C), thin dashed lines show where the distance limited the search.

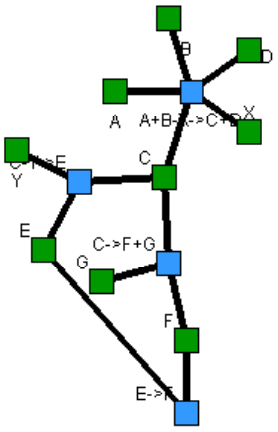

Figure 2.16: Graph layout for the test network.

By selecting another output format, the pathway query result can be transformed into a graph layout. Fig. 2.16 shows the result of such a transformation for our example network. Here the loop between C, E and F can be seen, and the general structure of our net is directly evident. In cases with more nodes, the name labels start to overlap and quickly become unreadable.

Tools like Otter [Huf00] alternatively allow us to show the name and additional information about a node by moving the mouse pointer over it, and they allow zooming in on regions of interest or picking other start nodes as centers for the layout. They can color nodes and edges based on their attributes. For example, molecule and reactions are assigned different colors here.

To show some examples from the actual database, Fig. 2.17 shows the environment around the Ras molecule for a distance of four and of eight. One can easily recognize the smaller environment in the larger one. In the interactive tool, one can make out important players associated with Ras directly, as they are linked to a large number of nodes: in both networks, more easily made out in the smaller one, the half circle to the right above Ras centers on Raf, the large one to the right below Ras on PI3K, and the one to the left below Ras on Grb2. In the larger network, the half-circle to the far left centers on p38, and the largest one, in the middle below, on NF-κB.

Inheritance

The user constructing pathways should be able to set the pathway query parameters to include subclasses and states, so that he will find a reaction even if he starts the search at a family or basic molecule that has no reaction linked to it. The same holds true for including superclasses, and finding reactions linked to them.

If Molecule entries in the database act as classes, we should think about inheritance of properties. Supporting inheritance means that the query and display mechanisms for the database have to find inherited properties and implement overriding. This behavior is similar to programming language classes, where subclasses can override behavior of their superclasses (see 1.3). As discussed in 4.2.4 we treat a Family entry in the database like a class, and its

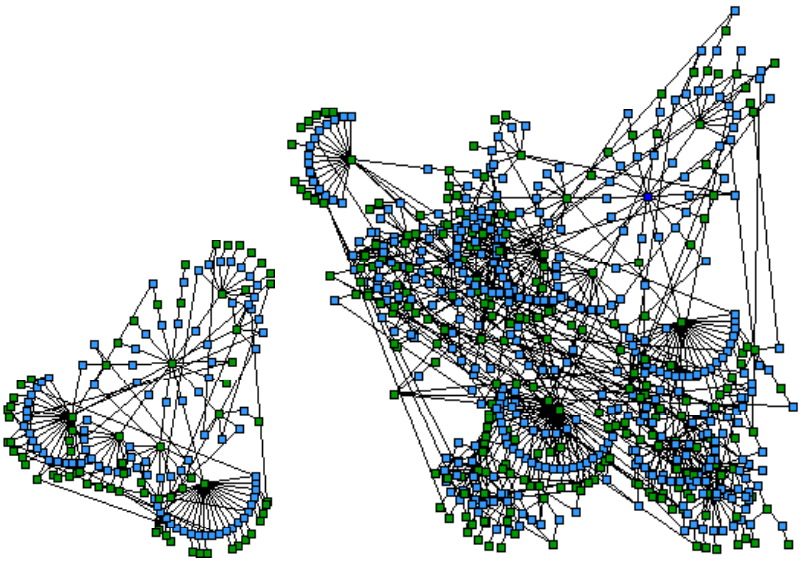

Figure 2.17: Graph layout for the network around Ras, with distances of four and eight steps, respectively. Ras is the element in the center of a circle of surrounding nodes in both images.

reactions and states represent its behavior. The analogy demands that our pathway engine suppresses the reactions or states of the superclasses when there are reactions or states for a subclass, and inherits them when there are none.

For example, suppose the general AC entry from Fig. 2.4 is linked to a reaction which states it is activated by $G_{s\alpha}$, and the calmodulin-dependent subtype ACI is linked to a reaction which states it is activated by $G_{s\alpha}$ in presence of calmodulin. Without overriding, ACI inherits the calmodulin-independent activation reaction *in addition* to the calmodulin-dependent activation — which is wrong.

The current implementation of the pathway builder does not support inheritance. Instead it explicitly shows to which entry reactions are linked.

By analogy to the search direction and distance, to include family relationships we can set a *range* and a *depth* for the search. If the range includes

superclasses, the builder follows the family edges *along* the direction in which they are pointing.

subclasses, the builder follows the family edges *against* the direction in which they are pointing.

super- and subclasses, the builder follows the family edges in both directions *at any node*, so that the whole connected family net around the starting node is traversed, similar to the reachable option.

Super- and subclasses are evaluated at any node only after all other pathways branching off in upstream or downstream directions have been exhausted. The depth works for the range as the distance does for the direction.

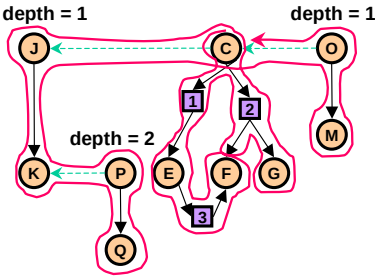

Figure 2.18: Traversal for the test network with super and subclasses, depth 2, downstream of C, any distance.

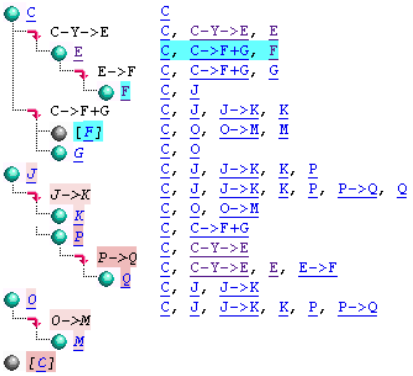

Figure 2.19: Pathway for the test network with super- and subclasses, as cascade and as list of shortest paths.

Fig. 2.18 shows the cascade that is generated for the test network with both super- and subclasses included. The pathway builder shows family transitions by coloring the background depending on the depth from the original level. The figure on the far right shows the list of the shortest paths to each node in the cascade, which the pathway builder can generate as an alternative view.

Filtering and Bounding

The pathway builder works by traversing the graph. The *search engine* by itself does not execute traversal-unrelated actions upon hitting a node. To restrict the traversal or define other actions to be executed on the node, *listeners*[7] can register for the search. The listeners form a pipeline: whenever a node is hit, an *event* is dispached to the first listener. The listener evaluates the event, may add a message for the following listeners to it, and passes it on. Each listener also can send callback messages to the search engine to make it skip a subtree or to stop the search. This design pattern is flexible and allows us to plug together a search from modules.

An additional set of listeners is called after the search has finished. This allows actions which are based on all the information seen during the search.

There are three classes of listeners that have been implemented:

1. *Boundaries* limit the search at nodes based on their internal criteria. For example, restricting the traversal to a certain distance is done by a listener which knows the distance limit and sends a signal to skip when the distance of the current node is larger.

2. *Filter listeners* select nodes based on their internal criteria. They are used to highlight interesting parts of the output. For example a filter listener can be used to highlight all transcription factors in the pathway.

[7]Listener is a technical term from the design pattern [GHuJV86] community. It means a piece of code that can be registered with another piece of code, to be *called back* when certain conditions in that other code occur.

3. *Print listeners* use *Formaters* to transform the nodes into some kind of output format for visualisation. For example, a print listener is used to produce the graphical cascades, another one transforms the pathway into an input file for the graph layout engine, and a third one transforms crosstalks found between two pathways into the format for AliBaba2 [Gra00], a tool that finds transcription factor binding sites in promotor sequences.[8]

Result set management

Filters and boundaries both select nodes. Therefore, any listener that selects nodes can be used either as a filter or as a boundary. A general solution to the selection problem is to explicitly enumerate the set of nodes we want to select. This is supported by the interface, which allows storage of any result set of a query or pathway query.

There can be specialized listeners for cases where the POET implementation of OQL is not expressive enough to generate the sets. For example a listener that selects nodes with a minimum number of connections has been implemented.

The set manager supports new queries within result sets to allow iterative focussing on the entries of interest. A set can be edited, for example added to an already existing set, so that the new set is the union of the two original sets.

Those sets then can be used as selectors in new pathway queries. For example, one could select all molecules and reactions located in rat and limit the search to this set, or exclude this set from the search.

2.5.5 Flatfile generation

Flatfile generation makes use of a Formater interface for printing the entries that result from an OQL query or from the result storage. There is one implementation of the interface per output format. It is possible to output the results of the web interface query directly into a file, currently as HTML, XML, plain text, Odf data for graph display, and AliBaba2.

2.5.6 Parsers and data import

The import of data from flatfiles is implemented with input file specific parsers, which offer a common Flatfile interface to the database. The XML parser is the default parser for the database. Fig. 2.20 shows an overview of the library for parsing the data. The web interfaces or commandline-tools and applications are front ends to this library.

10.073 Molecules in the Transpath database are imported from SWISS-PROT public Release 36. The high number results from inclusion of several vertebrate species. Records in the input file are translated into sets of objects referring to each other. For example, a typical SWISS-PROT entry is translated into a Molecule, several Hyperlinks, several Locations, several Annotates and several References.

In object oriented databases objects point directly to each other. A two-pass strategy is employed to import sets of related objects. In the first pass over the input file, entries without linking information are created in the database. In a

[8] The idea here is to see if co-binding factors are co-regulated.

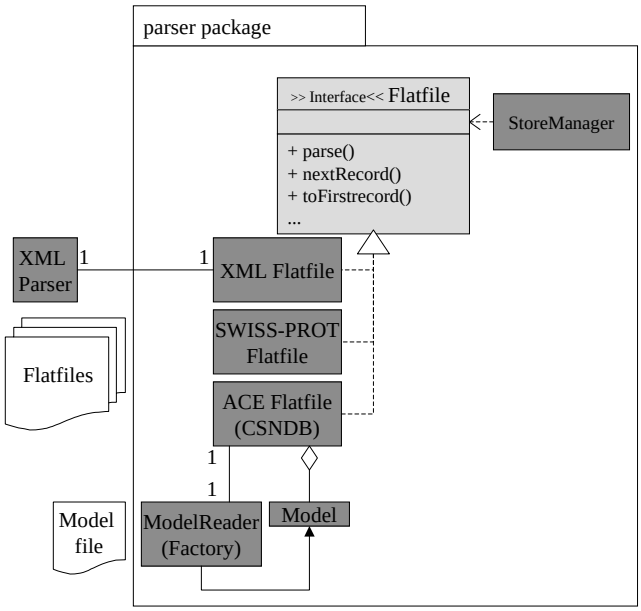

Figure 2.20: UML layout of the parser package.

second pass, the entries referring to each other in the input file are linked in the database.

Every imported object is uniquely identified by storing its source database and accession number in a "external identifier" field for its imported entries.

The contents of the CSNDB [TIK00] database on signal transduction are available in a separate database instance, which can be selected as an alternative data source under the Transpath interface.

CSNDB uses a public DBMS called Acedb[DM91], which allows storage of user-defined complex data types. The layout of these types is defined in a "model" file, which also declares name tags for the property slots. The data are distributed as a flatfile marked with those tags.

The class layout of CSNDB as well as that of Transpath changes over time, as improvements are realized. To decouple the class layouts from each other, a ModelReader uses a file with mapping rules to produce Model objects, one object per CSNDB class. The Model objects describe how every property of the CSNDB-class is translated in the Transpath schema. If either class layout changes, only the map file has to be updated.

Chapter 3

APPLICATIONS

Hardware lives for a few years, software lives for many years, but the data lives forever. The value of any database lies in the stored data. The presentation and storage systems for this data may change, but the data is there to stay.

The human mind can hold about seven pieces of information and their relationship at any one time [Voi00]. Valuable information is often hidden in large amounts of collected data and computers have to be employed to extract it. Data Mining [WF99] is a whole research discipline concerned with finding interesting patterns in data and clustering data into groups for apprehension by the human mind.

A good example are the DNA sequence databases. The human genome sequence holds most of the information that is necessary to create human life, yet by looking at it with the unaided eye, we understand nothing. In the wake of the accumulation of DNA sequence data, a host of algorithms has therefore been developed to aid in analysing and handling this data. Gene-finding software separates coding regions from untranscribed regions. Sequence alignment software like BLAST [AMS+97] makes it possible to find similar sequences and quantify their similarity. Evolutionary trees are constructed from such similarity calculations.

Signal transduction data can likewise be used as input for algorithms. Better descriptive overviews, analyses or even simulations are imaginable. The development of suitable algorithms provides many challenges for research. As first steps in this direction, three simple applications based upon the kind of data stored in Transpath have been devloped.

The first of these applications is an examination of the static connectivity of the network, the other two are dynamic simulations using models of signaling pathways.

3.1 Path alignments

A signal transduction pathway for our purposes in this section encompasses the molecular interactions triggered by one or more extracellular signaling molecules that lead to the activation of one or more transcription factors.

One of the questions regarding such pathways is: where is the specificity achieved?

Path alignments are employed here to suggest a possible answer. The idea behind path alignments is simple: we reduce the pathway network to sequences of molecules. Then we use sequence analysis tools to explore the relationships between these sequences.

3.1.1 Paths

For the following discussion we again represent the signal transduction network as a directed bipartite graph, where the nodes are the network's molecular species and reactions as described in 2.4.2. The molecule nodes are labeled according to their class. In particular, we have among them a set consisting of extracellular signaling molecules S_i and a set of transcription factors T_j.

Instead of dealing with whole pathways directly, we split them into bundles of linear paths. Properties of the pathway that rely upon the integration of two or more input signals or on feedback effects are thus lost in the new form. We define

Definition 1 (Signal transduction path) *A signal transduction path is the cycle-free series of edges and nodes leading from a molecule node labeled as extracellular signal to a molecule node labeled as transcription factor.*

Such a path is linear and can be represented as a string of nodes. Consider for example the simple network in Fig. 3.1: In this network there are three paths. When we skip the reaction nodes, which for our purposes do not carry additional information, we can write them as $P_1 : S_1/A/B/T_1$, $P_2 : S_2/H/I/T_2$, and $P_3 : S_1/H/I/X/B/T_2$. The slashes serve as delimiters of the node names.

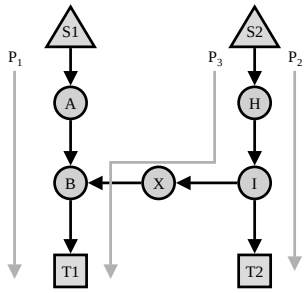

Figure 3.1: Short arrows represent reactions, S_1 and S_2 extracellular signaling molecules, T_1 and T_2 transcripton factors, A, B, H, I, and X intermediary molecules. P_1, P_2 and P_3 are paths.

3.1.2 Assumptions

We assume that the shortest path between an extracellular signaling molecule and a transcription factor is the most efficient and biologically relevant path, as it is wasteful and more error prone to use a longer path.

3.1.3 Method

Based on this assumption we filter out the shortest paths between all possible pairs (S_i, T_j) present in the database, by performing a breadth-first search for each S_i and retaining all paths that lead to one of the T_j. Each of these paths may share part of its sequence with other paths. In our example the path P_3 does this with P_1 and P_2.

Since paths are sequences of symbols, we treat them in the same way as we treat nucleotide or protein sequences, and we use alignment tools to compare them. We also use the similarity scores generated by such alignments to cluster them into hierarchical trees, where similar sequences show up in neighboring leaves.

The difference between path and polymer sequences is that for paths we have to deal with an alphabet of thousands of different symbols, whereas for polymer sequences there are only four different nucleotides or twenty amino acids. Common bioinformatics alignment tools like ClustalW [THG94] cannot be used for our purposes as they exploit the limited number of letters in these alphabets to be more efficient. ClustalG [WH99], originally developed for event series analysis in psychology, is a general alignment tool which does not have these limitations.

When matching two nodes against each other, the simplest comparison regime is an identity test. A score of one is assigned if both nodes are identical, otherwise zero is assigned . Ideally, we should reflect similar signaling behavior between members of a protein family by assigning a positive similarity score to a match between two family members.

ClustalG presently does not support similarity matrices. Therefore we use the identity test.

Generating the paths

A breadth first variant of the graph traversal algorithm allows listeners to register and track the shortest paths. Listeners exist for hyperlinked HTML, plain text and the FASTA format employed by the ClustalG alignment tool. The plain text output for the path from acetylcholine to NF-κB looks like this:

> acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB: 10

The number at the end designates the number of steps. Slashes are used instead of hyphens to avoid ambiguity with hyphens in molecule names. For more detail into where the family transitions occur, it is possible to print the reactions, too.

The FASTA input format for ClustalG consists of a name for the sequence and a five letter string for each letter in the alphabet. We use the ends of the sequence, the extracellular signaling molecule and the transcription factor, to name the sequence. This name is unique since only one shortest path between the two nodes will be found. We convert the entry accession numbers to base 26 to get letters, and the above path looks like this (all generated paths can be found in in Appendix C):

>acetylcholine@NF-kB—10
MaaezMadtyMadigMabswMaaloMaafdMadgwMacujMadxkMadrq

3.1.4 Results

We use the whole CSNDB-dataset to create all possible shortest paths, exept for the nuclear receptors. In the case of the nuclear receptors, the path only consists of the two end nodes and an alignment does not make sense. The full alignment is included in Appendix C. Clustering the alignments and visualizing the distance tree in TreeView[Pag96] produces figure 3.2.

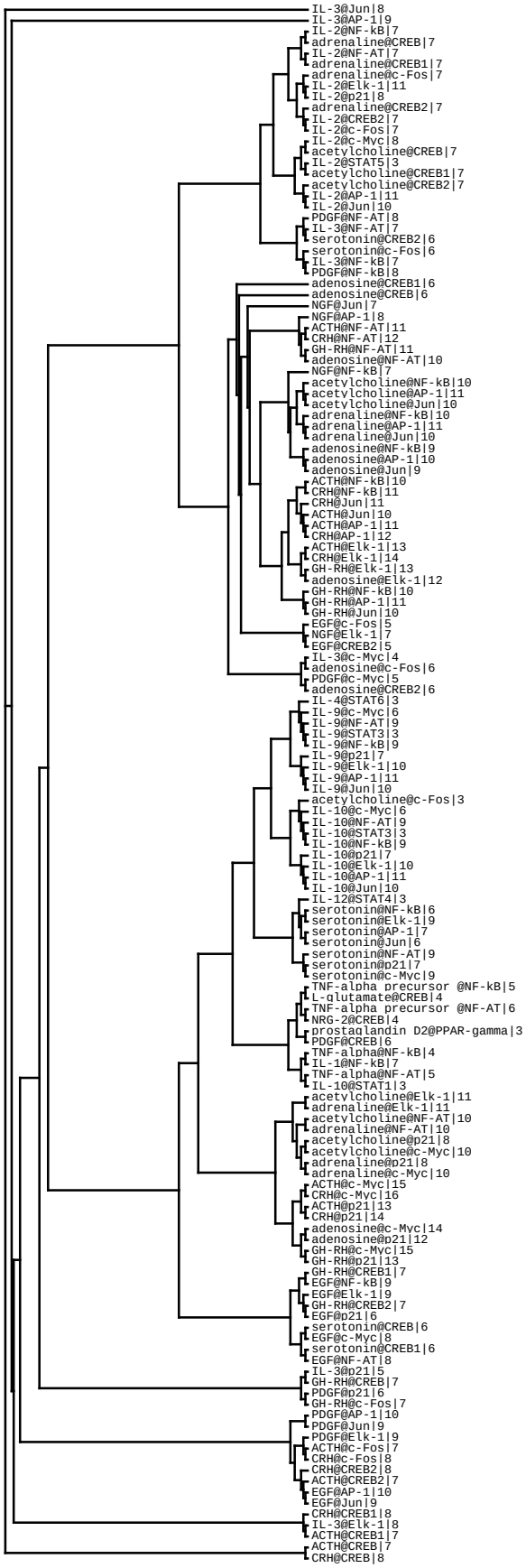

Figure 3.2: Distance tree for all paths in CSNDB.

3.2 Qualitative simulation with Boolean Networks

Facing sparse kinetic data, from the several possible methods to simulate network behavior, Boolean networks and Petri nets were chosen for investigation. Boolean networks are simple and modest in data needs, and Petri nets seem promising because they support different levels of detail. This section deals with the results of a simplified Boolean network for the MAP kinase cascade, and the next section will present a more elaborate model for the same cascade using Petri nets.

3.2.1 Boolean networks

Boolean networks [Kau93] are directed graphs, they consist of one node type connected by directed edges. The *wiring* corresponds to the edges in the graph. A single node has one or more *input* nodes, and can serve as an input to any number of *target* nodes. In the context of a single node, *wiring* refers to the edges coming from the inputs. Each node can be in one of two states, either "on" or "off". Usually on is numerically represented as a 1 and off as a 0. A rule table maps input states to a new node state. For a node with k incoming edges, the ruletable thus has 2^k entries, and the rule as many bits. The sum of all node states is the system state.

Starting from an initial distribution of on/off states, all nodes are evaluated and are updated synchronously, leading to a new system state. This *step* is repeated until a single final state or a cycle of repeating states is reached. Several starting conditions can lead to the same final state or cycle, which is the reason for calling these final states *basins of attraction*.

3.2.2 Modeling assumptions

Boolean networks can represent semantic or mechanistic reaction networks. As an example we model a semantic network, because in this kind of network each molecular species can be in only one of two states, either active or inactive, which corresponds naturally to the two states for a node in a Boolean network. The semantic information in the reactions can be captured in the wiring and the rule table.

We translate each molecule species into a node. The activation profile for each molecule species is assumed to be discrete: if the node is off, the species is fully in the inactive state: if the node is on, the species is fully in the active state. Intermediate levels are not supported.

We translate each reaction into edges and changes in the ruletable of the associated target nodes. If a reaction has several signal donors or acceptors, it would be a hyperedge in the graph, and is decomposed into several edges. If it only has one donor and one acceptor, it directly translates to a single edge in the graph.

The ruletable for each of the acceptor or target nodes is composed from the rules for each of the incoming decomposed reactions, according to the following scheme:

Any reaction is then exactly enabled, when all input nodes on the corresponding edges are on. In our example model, each reaction has only one corresponding edge and is enabled if this edge's input node is on. An enabled

activating reaction tries to switch on the target node in the next step, while an enabled inhibiting reaction tries to switch it off.

If activating and an inhibiting reactions compete for the same target and in a Boolean network, only one of them can win. We assume that the inhibiting one wins. Extending this notion to several activating and inhibiting reactions, the target is switched on only if there are more activating reactions than inhibiting ones.

3.2.3 Modeling procedure and parameters

Fig. 3.3 shows the layout of the network. The network demonstrates the canonical transcription activation pathway for the Ras-MAP kinase cascade after stimulation with EGF. It notably contains one positive feedback cycle, leading to the reactivation of PKC, and one inhibitory interaction by MKP. It does not contain any of the mechanisms to switch off the original signal that are present in the cell. Neither does it contain downstream effects and gene regulatory networks of the activated transcription factors, which would warrant a model of their own.

This network was stored in the database and automatically converted into input files for DDLab, a software for the analysis of Boolean networks (see Appendix A), a process that can be applied to any subnet of the database. Nodes without inputs are automatically wired back to themselves and so retain their state during the whole simulation. This is the case for EGF in our example, which of course does not have a self-activating reaction in the database.

The central dynamical simulation assumes all molecular species except for EGF, the signal of interest, to be inactive at the beginning. After we run the simulation, we can follow the cascade of activation events downstream of EGF.

3.2.4 Results

Fig. 3.4 shows the results of the analysis. Since the network is small, it is possible to enumerate all 2^{14} possible network states and achieve an exhaustive, static result for all possible outcomes. Larger networks, which are sufficient for practical problem sizes, restrict us to define starting conditions and then simulate until a basin of attraction is reached, as discussed above. There are statistical methods to estimate the number of basins and other properties even for these larger networks.

Here we have four basins of attraction, three resulting in a steady state and one resulting in an oscillatory cycle.

The first basin (upper left) is trivial, consisting of an empty network plus the combinations of MKP, the MAP kinase cascade and the Transcription Factors which lead directly to an empty network, empty meaning all species are in the "off" state.

The second basin (upper right) shows a weak activation of the feedback cycle, which persists even in absence of the original EGF signal, oscillating between on and off for all its members. The oscillation is caused by the inhibitory effects of MKP. Table 3.1 lists the states of all components during the four-step oscillatory cycle.

The third basin shows strong activation of the feedback cycle, again in absence of an external signal from EGF. Here the combined activating effects of

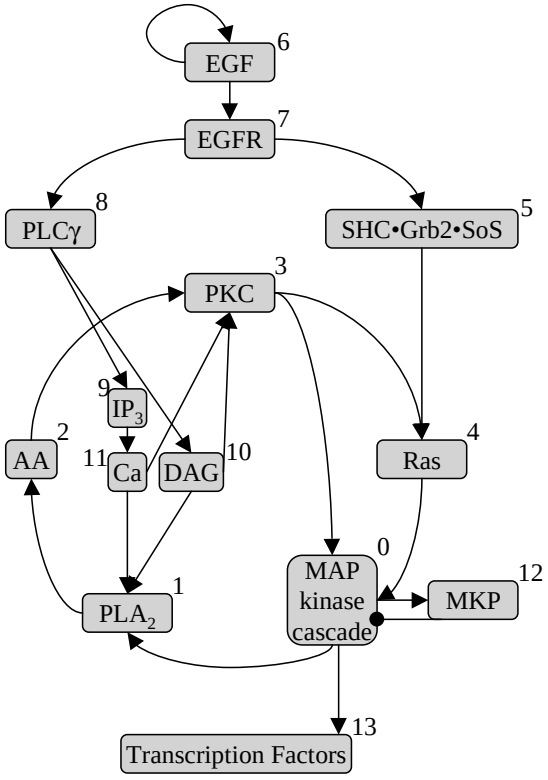

Figure 3.3: Block layout of the Ras–MAP kinase cascade. Pointed arrows: activation. Arrow with round end between MAP kinase cascade and MKP inhibition. The numbers correspond to the index of the node.

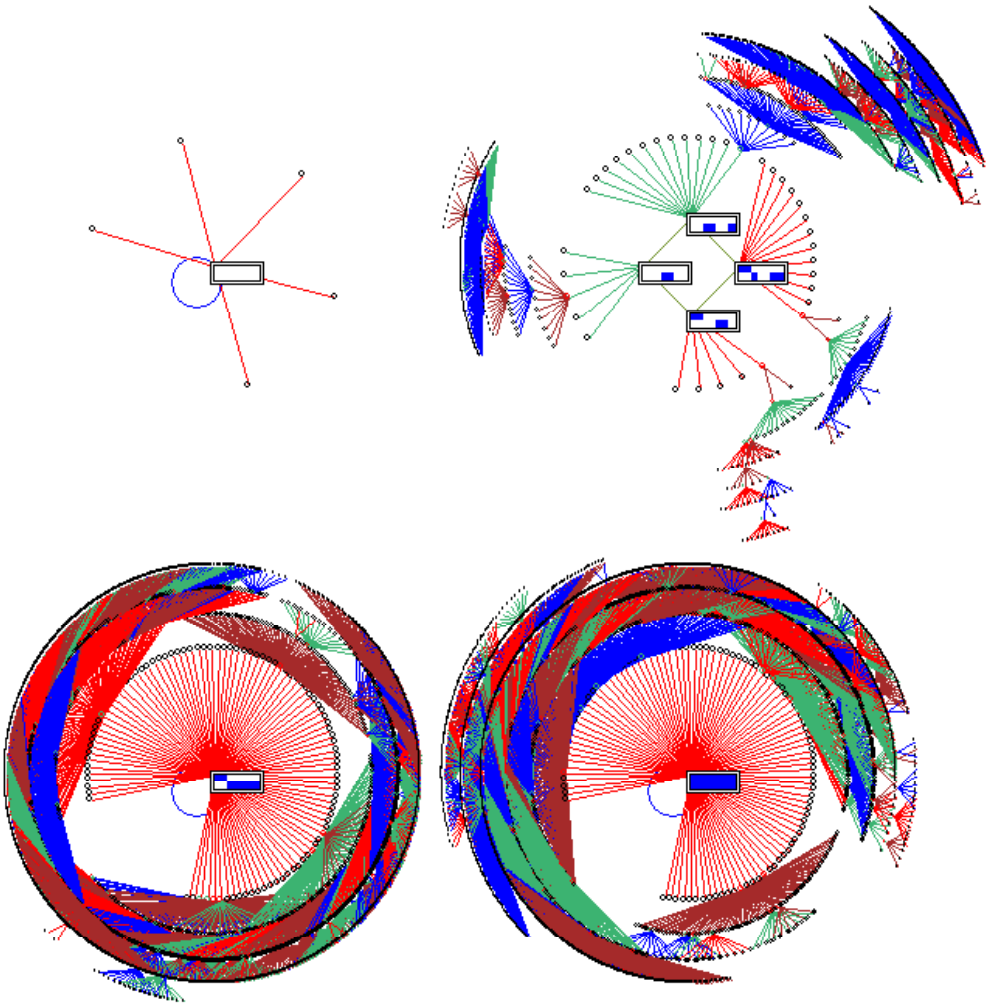

Figure 3.4: Attractor basins for the Ras-MAPK cascade. Each dot or circle is one of all possible states of the system. Circles on the outside are so-called "garden-of-eden-states" which can not be reached from any other state. Circles in the center represent the basin of attraction, and are labeled with a bit pattern representing the on/off states of all nodes. This pattern is broken up into sucessive rows, starting with the highest node index (13) in the upper left corner, and ending with the lowest node index (0) in the lower right corner. Successive system states are connected by lines. Overlaps of these lines are an artifact of the graphical representation and do not mean these states are connected. Also, the different hues of these lines only serve to make the grouping more clear and have no further meaning. The attractor basins are a static representation of the system change over time. From any state, a timestep leads the system to the next state, which is placed nearer towards the center of the basin and inidicated by the connecting line. Finally, when the center is reached, states proceed clockwise from one to another or a single state is repeated *ad infinitum*.

		State Nr.			
Index	Name	1	2	3	4
0	MAPK cascade	•			•
1	PLA_2	•	•		
2	AA		•	•	
3	PKC			•	•
4	Ras	•			•
5	SHC/Grb2/SoS				
6	EGF				
7	EGFR				
8	PLCγ				
9	IP_3				
10	DAG				
11	Ca				
12	MKP	•	•		
13	Transcription Factor	•	•		

Table 3.1: Active states during the oscillatory cycle. The first state corresponds to the right hand bit pattern in Fig. 3.4. It leads to the second (lower bit pattern), which leads to the third (left hand bit pattern), which in turn leads to the fourth state (top bit pattern). The fourth state leads back to the first one. • corresponds to "on".

PKC and Ras are stronger than the inhibiting effect from MKP.

The final basin (lower right) corresponds to all networks states where EGF is active. Since there is no mechanism in the model to switch off the EGF activation, any state where EGF is active will lead to full activation of whole network, including the feedback cycle. This basin contains the case mentioned above, activation of the network by EGF.

3.3 Quantitative simulation with Petri Nets

3.3.1 Petri nets

Basic Petri nets are bipartite, directed graphs. One node type, the *place*, represents an object in the net and can hold an integer number of *markings*. The other node type, the *transition* represents the actions between the objects. Places and transitions are connected by weighted, directed *arcs*. If no weight is assigned explicitly, the weight is 1. Places pointing to a transition via an arc are its *preconditions*. Places pointed to by a transition via arcs are its *postconditions*. A transition is called *enabled*, when all its preconditions are fulfilled, which means that they contain at least as many markings as the weights of the arcs.

For example, consider a transition that is connected to two preconditions. Let the connecting arc for the first precondition have a weight of one and the connecting arc for the second precondition have a weight of three. Then the transition is enabled whenever there is at least one marking on the first and at least three markings on the second precondition.

From all enabled transitions one is picked at random and *fires*, removing the markings from each of its preconditions and adding a number of markings to each of its postconditions, again according to arc weights.

In *timed* Petri nets, transitions have a time delay between being enabled and firing. They have to stay enabled during the whole delay. The transitions are also called *activities* in these nets.

The goal is to evaluate the general usefulness of Petri nets as support tools for the data analysis and visualisation of cellular signaling data. One objective is to recreate a detailed simulation of the process and then try to see how robust the model would be to parameter loss.

3.3.2 Modeling assumptions

The model for the quantitative simulations is chemical reactions, either unidirectional catalyzed or bidirectional equilibrium reactions. We use chemical kinetics to describe the rates of those reactions.

Chemical Kinetics

Detailed simulations trace the concentration changes of signaling molecules. Chemical reactions produce these changes, and chemical kinetics is the theory that mathematically describes the rate of concentration change for these reactions. The rate of change depends mainly on the rate constants for the reaction, and on the concentrations of the chemical compounds, temperature, pressure, and on the presence of catalysts or inhibitors. In biological systems, pressure and temperature change little and they are consequently hidden in some constants. For the description of the rate of enzyme catalyzed reactions the Michaelis-Menten [Hal30] model is commonly used.

The equations for an unidirectional and an equilibrium one-step chemical reaction are

$$\sum_{i=1}^{N} \nu_i X_i \xrightarrow{k_f} \sum_{i=1}^{N} \nu_i' X_i$$

and

$$\sum_{i=1}^{N} \nu_i X_i \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \sum_{i=1}^{N} \nu'_i X_i$$

where v_i and v'_i are the stoichiometric coefficients of the reactants and products, X_i are the chemical species, k_f and k_b are the forward and backward rate constants, and N is the number of species. The unidirectional reaction is an idealization for reactions which have backward reaction rates that are small enough to be neglected.

Reaction rate for equilibrium reactions

The *extent of the reaction* ξ is the stoichiometry-corrected change in the amount n_i of a substance: $\mathrm{d}\xi = \frac{\mathrm{d}n_i}{\nu_i}$. The *conversion rate* is defined as $\dot{\xi}$. The *reaction rate* υ is the conversion rate divided by volume V, thus expressing the rate depending on concentration change. We can also calculate the reaction rate directly from the concentrations:

$$\upsilon = \frac{\mathrm{d}[X_i]}{\nu_i \mathrm{d}t} = k \prod_{i=1}^{N} [X_i]^{\alpha_i}$$

where k is the *rate constant* for this reaction[1] from and α_i is the *order* of the reaction in respect to the reactant X_i. The overall order of the reaction is $\sum_{i=1}^{N} \alpha_i$. A typical unit for the reaction rate is $moles/l \cdot s$, where l stands for liter and s for second, often noted in short as M/s, where M stands for *molar*, or $moles/l$. The unit for the rate constant is $M^{1-n}s^{-1}$ for an n-th order reaction.

Integration of this equation to predict the state for the system at a given timepoint is possible for simple cases, like unidirectional and reversible first and second order reactions or consecutive or parallel first order reactions. For complicated networks like the ones we deal with in signal transduction, only numerical analysis is feasible.

In elementary reactions, which cannot be broken down into several sequential steps of simpler reactions, the reaction order of a reactant equals its stoichiometric factor or $\alpha_i = \nu_i$, and the overall reaction order of all reactants equals their *molecularity*.

[1] This constant can be derived by consideration of microscopical and thermodynamical properties. For a bimolecular reaction between X and Y it is given by the *Arrhenius equation*:

$$k = \sigma_{XY}^2 \sqrt{\frac{8\pi KT}{\mu}} \underline{P} \cdot e^{-\frac{E_a}{R_u T}}$$

where σ_{XY}^2 is the collision radius of the two compounds, K is Boltzmann's constant, T is the absolute temperature, μ is the reduced mass $m_X m_Y/(m_X + m_Y)$ with m_X and m_Y being the respective molecular masses of the two molecules. This first part represents the probability of a collision. $\underline{P}$ is a steric factor accounting for the relative orientation of the reactants. The exponential *Boltzmann factor* accounts for the fraction of collisions with energy greater than the necessary activation energy E_a. R_u is the *Universal Gas Constant*.

Reaction rate for catalyzed reactions

These are the proposed reaction mechanism and the rate equation for the Henri-Michaelis-Menten model:

$$\mathrm{E} + \mathrm{S} \underset{k_b}{\overset{k_f}{\rightleftharpoons}} \mathrm{ES} \overset{k_p}{\rightarrow} \mathrm{E} + \mathrm{P} \qquad\qquad v = V_{max} \frac{[\mathrm{S}]}{K_m + [\mathrm{S}]}$$

S, E, P, and ES are the substrate, free enzyme, product, and enzyme-substrate complex. V_{max} is the maximum reaction rate, K_m is the Michaelis-Menten constant defined as

$$K_m = \frac{[\mathrm{E}][\mathrm{S}]}{[\mathrm{ES}]} = \frac{k_b + k_p}{k_f}$$

This model for catalyzed reactions is based on several assumptions, namely that

1. Only a single enzyme and a single substrate are involved. The enzyme has only one active site. $[\mathrm{E}]_{total} = [\mathrm{E}] + [\mathrm{ES}]$.

2. The substrate is saturating so that the formation of the ES complex does not change the substrate concentration, and nearly all enzyme is in complexed form. $[\mathrm{S}] \gg [\mathrm{E}]$, $[\mathrm{ES}] \approx [\mathrm{E}]_{total}$.

3. The *Briggs-Haldane steady-state* approach assumes that a short time after the start of the reaction, ES will build up as fast as it decomposes and keep a steady concentration level until near the end of the reaction.[2]

4. Catalysis is the *rate limiting* step. $V_{max} = k_p[\mathrm{ES}] \approx k_p[E]_{total}$

Simplifying Assumptions

For our model, we assume that all reactions are elementary. This simplification is false in many cases, but allows us to model the network even if the true reaction orders are not known.

Likewise, we assume that catalyzed reactions follow Michaelis-Menten kinetics. This also is questionable, as many substrates *in vivo* are not saturating but show concentrations in the same order of magnitude as the enzyme itself. Also, several enzymes may share substrates, and several substrates may be valid for a single enzyme.

In addition to these difficulties, since many signaling molecules have low copy numbers, local concentrations in the cytoplasm can vary significantly.

A stochastical approach, which does not assume that the molecules are homogenously distributed in the cell might be preferable. Purely stochastic simulators have been developed [Gil77] for this purpose, but are not employed here.

We employ a special kind of Petri net, called a *stochastic activity net* to account for the fluctuations of concentrations and reaction rates in the cell. A

[2]The original Henri-Michaelis-Menten model uses $K_s = k_b/k_f$ instead of K_m and requires that enzyme, substrate, and enzyme-substrate complex are in rapid equilibrium, which means $k_b \gg k_p$. This assumption is called the *quasi equilibrium* assumption, and is a special case for the steady state formula.

probability distribution represents delays in such a net. All delay probabilities in our model use the exponential distribution, which means that for a given rate λ, the *average* time for a reaction to happen equals $\frac{1}{\lambda}$, but individual times may vary.

Many enzymes are regulated or inhibited by competition, covalent modification, or allosteric effects. These circumstances were modeled by representing the explicit mechanism of inhibition or modification.

3.3.3 Modeling procedure and parameters

This section provides an overview about the modeling. A full list of all parameters for the nets can be found in Appendix D. Parameter values were taken from the paper of Bhalla and Lyengar [BI99], as well as from the literature cited therein.

Run pooling

The stochastic fluctuations in reaction rates lead to rugged changes in the concentrations. Each of these rough curves represents one particular instance of activation of the signaling pathway. To abstract the average behavior of the pathway, we have to run the simulation several times and average the resulting curves.

The times for the firing of activities differ from run to run. If we have two runs, let us consider a time-step in the first run, which does not occur in the second run. To average the concentrations at this time-step, we interpolate the concentrations in the second run linearily (See Fig. 3.5).

One run will be longer than the other. We extrapolate the value for the final timestep in the shorter run by repeating the last timestep. We do not extrapolate using the incline between the last two measured points in the shorter run. If we did, we would exaggerate the random fluctuation between of the last two time steps, because the difference between the length of the shortest and the longest run can be large, which can lead to impossible negative values for the extrapolated concentration.

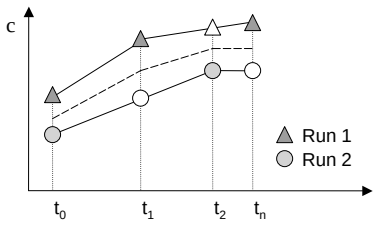

Figure 3.5: Run interpolation. White circles and triangles are interpolated values, filled ones are values generated by the model. The dashed line is the averaged curve.

Units

The concentrations of most signal transduction molecules are nM to μM. In the Petri net representation, integer numbers of tokens represent concentration.

To avoid fractions, all concentrations are therefore given in nM. Rate constants for k_f and k_b are given in $nM^{-1}s^{-1}$ for bimolecular reactions, rate constants for k_f, k_b, and k_p are given in s^{-1} for unimolecular reactions. K_m is given in nM.

3.3.4 Results

To demonstrate the properties of the modeling approach consider the classic enzyme-substrate mechanism model as discussed above, which does not make use of some of the simplifying assumptions needed for the Michaelis-Menten model. It is depicted in its Petri net representation in Fig. 3.6. This model exhibits unidirecional and equilibrium reactions, as they will be used in the signaling cascade model later on.

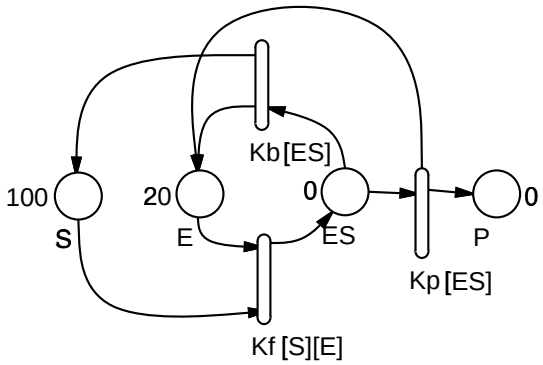

Figure 3.6: Enzyme-substrate mechanism Petri net. Circles are places and are labeled with their name, vertical bars are activities and are labeld with their rate function. Numbers represent initial place-markings in the net. $K_f = 5$, $K_b = 4$, $K_p = 1$

The results for a single run and for ten pooled runs can be seen in Fig. 3.7. The smooting effect of averaging is clearly visible. In the single run simulation, the substrate was transformed faster than it would be on average. The observed behavior for the pooled runs matches the theoretical expectations. Initially there is a fast equilibrium reaction and the enzyme is saturated with substrate. Then the expected steady state ensues, until the substrate concentration drops enough to approach the enzyme concentration. Finally all substrate is transformed to product and the reaction ends.

In the signaling cascade we employ the Michaelis-Menten model to simulate enzymes. The according Petri net is shown in Fig. 3.8.

The reason for this is that fewer places and activities are needed, and the Michaelis-Menten model fits to the more detailed mechanistic model closely enough.

Two interesting points can be observed upon comparing the two models in Fig. 3.9. First, the enzyme-substrate complex formation with the sharp drop in free substrate at the beginning is missing in the Michaelis-Menten model. This is expected, since the Michaelis-Menten equation assumes that the enzyme concentration is negligible compared to the substrate concentration. Second,

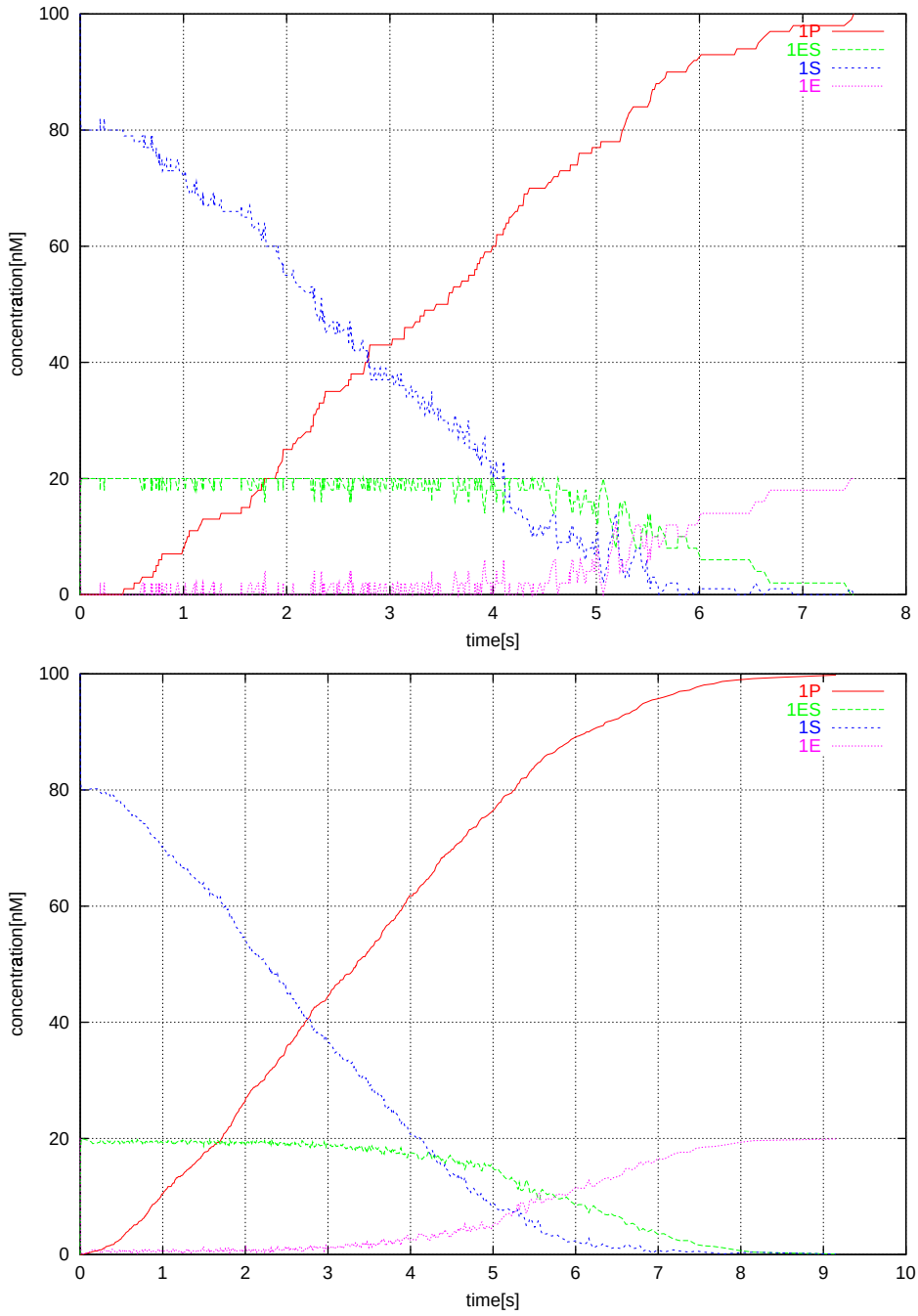

Figure 3.7: Results for the enzyme-substrate mechanism Petri-net. Upper image: a single stochastic run, lower image: average over ten runs.

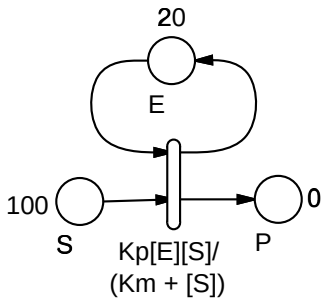

Figure 3.8: Michaelis-Menten enzyme Petri net. Symbols and constants are the same as in Fig. 3.6. $K_m = K_b + K_p/K_f$

the Michaelis-Menten model completes faster than the detailed model. The reason for this is that the Michaelis-Menten model assumes steady state during the whole process, and cannot reflect the slow-down when this assumption does not hold any more near the end of the simulation. The models fit well up to this point (at about six seconds in our example). Of course in the Michaelis-Menten model the enzyme concentration is constant.

After this demonstration of the basic applicability of modeling approach, we now try to extend it to modeling a more complicated network, namely the EGF-Ras-MAPK cascade. The cascade model is composed of three subnets (see Fig. 3.10), which are then combined into one overall net. Two of these serve as the biological model, while the third regulates the duration of the input signal.

The signal timing subnet works with the aid of an *output gate*. An output gate in a stochastic activity net changes the output behavior of a firing activity. In our case the clock fires on average every second. The output gate then increases the clock's number of tokens by one each time, and refreshes the pool of tokens for EGF. When there are more tokens on the clock than the time limit allows, it removes all tokens from the clock and from EGF, shutting the signal down.

Fig. 3.11 shows the results for an example of two pooled runs of this network with a 10-minute stimulus of 5 nM EGF.

The qualitative behavior of our simulation for the Ras-MAPK cascade (see Fig. 3.11) resembles the experimental result, with a temporary steep increase in MAPK-activity. This increase happens faster in the real cell, where the peak is after about 10 minutes instead after half an hour. Probable reasons for this difference are the simplifying assumptions about the reactions we used, differences between measured and physiological rate constants or missing reactions in our model.

The run also shows two other properties of the model:

First, the temporal behavior of the Shc*·SoS·Grb2 complex. The concentration of this complex first increases to a saturation plateau. After a delay introduced by the intermediary reactions, this leads to the formation of MAPK*. The negative feedback of MAPK* finally removes SoS from the equilibrium for the complex, and the complex concentration drops.

Second, after 10 minutes the EGF signal is switched off, and the subsequent internalisation of EGF-bound receptor shuts the signal down.

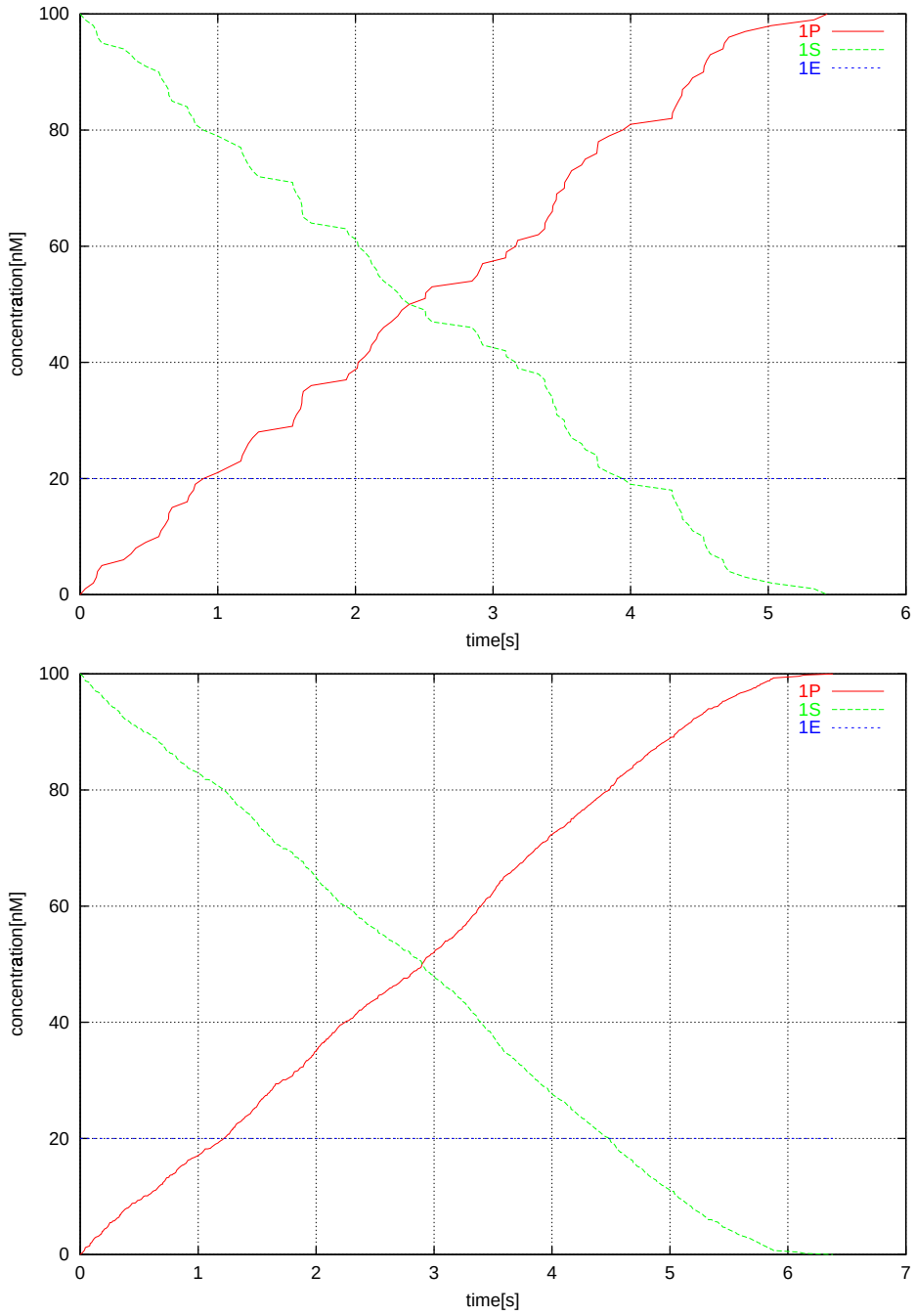

Figure 3.9: Results for the enzyme-substrate Michaelis-Menten Petri-net. Upper image: a single stochastic run, lower image: average over ten runs.

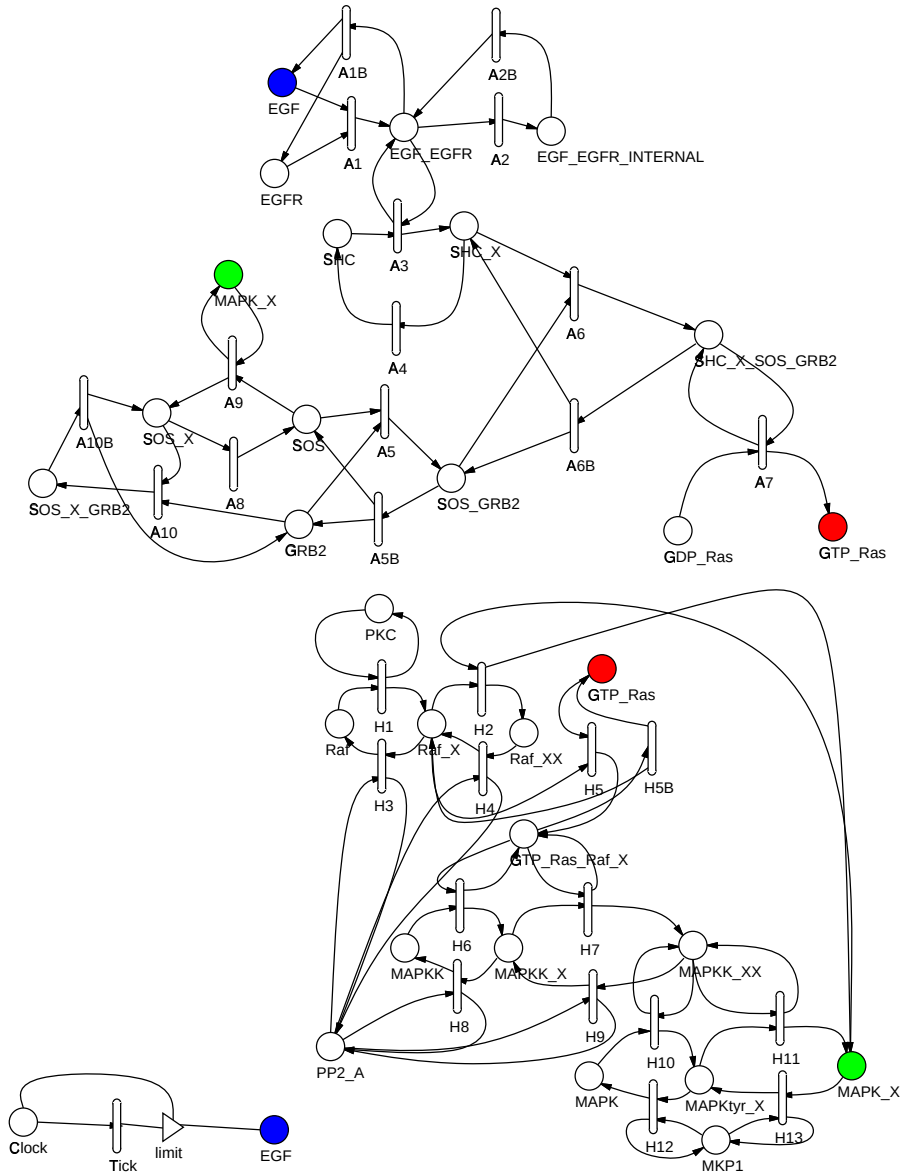

Figure 3.10: Ras-MAPK cascade Petri nets. The places EGF, MAPK_X (active MAP kinase) and GTP_Ras (active Ras) are shared between the subnets and serve to connect them. They are the filled circles in the image. The triangle is an *output gate*. Other symbols are the same as in Fig. 3.6

.

Name	Conc. $[nM]$
EGF	varied
EGFR	166
PKC	20
GAP	2
GDP·Ras	200
Grb_2	1000
MAPK	360
MAPKK	180
MKP1	3
PP_2A	224
Raf	200
SHC	500
SoS	100

Table 3.2: Starting pools for the simulation. Species not mentioned have a concentration of 0 nM. Parameter values were taken from the paper of Bhalla and Lyengar [BI99].

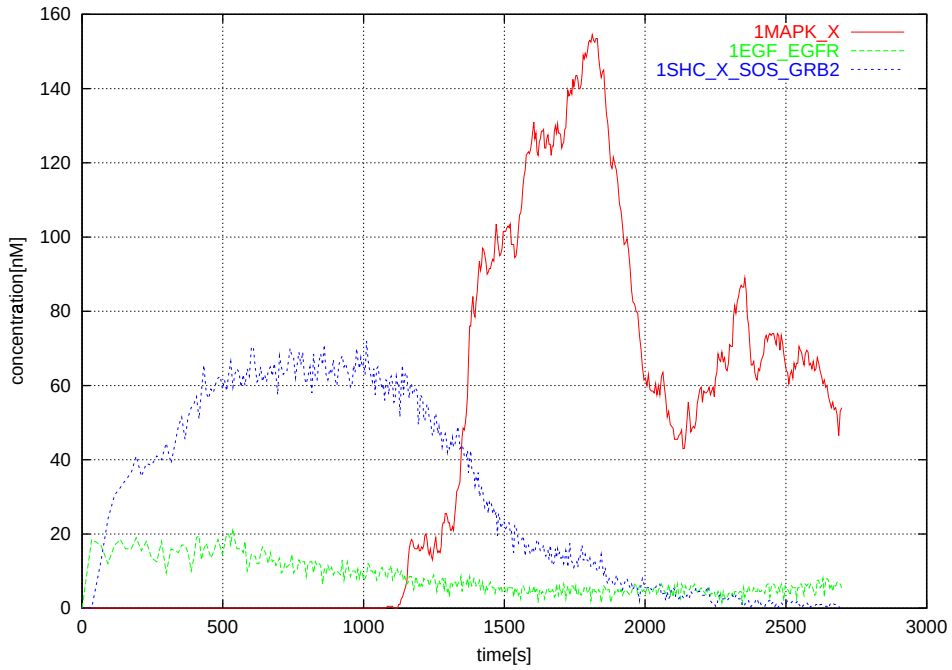

Figure 3.11: Results for the Ras-MAPK cascade combined Petri net for a single run with 10 minutes of 5 nM EGF stimulation.

Results vary quantitatively from run to run, but the qualitative behaviour of the network stays the same (data not shown).

Chapter 4

DISCUSSION

This chapter discusses the results that are presented in chapters 2 and 3 and points to further areas of investigation.

4.1 Data

What we want to have in the database is the accumulated worldwide knowledge about signal transduction as it happens in healthy, natural mammals.

Unfortunately, no one today knows what happens in healthy, natural mammals. In finding out about signal transduction, researchers are restricted to experiments that are conducted in specialized laboratory cell lines, with mutated genes inserted into cells by phages, with in vitro studies using protein obtained from bacterially expressed plasmids, with knock-out mutations and so on. This kind of knowledge is published in the seminal literature, together with reviews. That literature in turn is the deposit that we can mine for data to input into our database.

How do we extract data from published scientific literature? There are endeavors to automate this process by parsing the abstracts of biological papers, checking for names of molecules and keywords that relate them with each other, like "activates", "binds" or "downregulates". Natural language has a lot of irregularities, and to date, these works have not come up with a satisfying result, although the idea looks promising.

The most arduous process for obtaining data is undoubtedly extraction from scientific literature by hand. There are no real alternatives to this method if validated, high quality data is desired. Apart from an import of 10.073 molecule entries from the SWISS-PROT database [BA00], public release Nr. 36, which also is built by manual extraction, all data entries in the database have been obtained in this way.

We are careful not to mix overviews derived from this data with the data itself. Rather, overviews should be constructed from the hard, underlying data automatically and be marked as such. One example is a pathway, which has been elucidated in its different parts in different organisms. It should be possible to extract this pathway from the database, and it should always be possible to find out that it was stitched together.

The task is complicated by the review literature. On one hand it is desirable

to extract information from this literature, since the information has already been distilled into an overview by an expert. This allows us to enter considerable amounts of pathway quickly, which is important to get adequate coverage of the field in a realistic timeframe. On the other hand, this information already contains the kind of abstractions that we want to make possible, but keep separate from the data, and we enter it as data.

If we were strict, no evidence that has not been investigated experimentally should be entered into the database. Higher-level understanding constructed from facts could be shown as such. Since data is entered by hand and the annotators' time is limited, this approach is not practical. An annotator would spend years fleshing out all the details of a small part of the network, which would be a delight for specialists in this area and make the database next to useless for everyone else. For this reason, the first milestone for the data is to provide an overview of the cellular signaling system, a skeletal signaling network. Once this is in place, the strategy is to flesh out areas of the network according to demand.

The separation between semantic overview reactions and mechanistic detail reactions is valuable, as it helps to keep the two kinds of data apart.

In order of decreasing detail and modeling power, and of increasing availability, the following data can be retrieved from the scientific literature:

- Kinetic data: speed constants, equilibrium constants, and concentrations of molecules. These last of course would have to represent some steady state in a given cell and tissue type of a given developmental stage.
- Stoichiometric coefficients: they assign weights to the connections, which are needed to calculate steady state equations.
- Semantic data: describes the effect of an interaction, but does not quantify, for example inhibition or activation, or whether an interaction is direct or indirect.
- Connectivity data: describes which molecule interacts with which other molecules.

4.2 Database

4.2.1 Graph representations

One can represent the network as a graph with only one type of node, the signaling compounds, where the reactions are implicitly contained in the edges; or as a bipartite graph, where the reactions also form nodes; or even as a tripartite graph, where the connections between reactions and compounds form nodes. It may be convenient to view the bipartite graph as a hypergraph, where reactions are the hyperedges, edges with more than two associated nodes.

Many representations are possible. *The best representation for understanding the networks is the one that depicts the modeled entities and their reactions similarly to the way we think about them.* Since molecules and their reactions are the concepts we have for the signaling network, that is the form we chose in the database. We chose a form where the reactions skipped when the reactions are not of interest, as is the case for path alignments.

4.2.2 DBMS

The database could be implemented in any kind of database management system, be it a classical relational system, an object-oriented system, an object-relational system or an even more exotic format, such as a file with Prolog assertions.

The decision to use an object-oriented system over the better established relational systems was based on the following considerations:

- As the main point, the expected high connectivity between the data items (see 2.1 below) makes it improbable that the SQL interface to relational databases would be well suited for retrieving the data. To construct pathways out of single interactions, repeated joined queries for small amounts of data to the database would be needed. An object-oriented database with direct pointers between related objects was expected to handle this requirement much more gracefully than a relational system.
- Object-oriented databases allow direct use of the stored data items in the application language, avoiding the tiresome object-relational mapping process. It was expected that the development of the prototype database would be considerably faster due to this.
- Theoretically object-oriented databases are superior to relational databases. Some weaknesses of the relational system are that it represents complex data types only indirectly and cannot return them as results of queries, it does not directly support inheritance and does not support recursive querying of structures. Object-oriented systems allow us to represent biological objects more directly than relations would be able to do. Part of the project was therefore the evaluation of available object oriented database technology for biological databases.

Object-oriented databases are better for modeling and prototyping than relational databases. They are worse for integration of their data with relational data.

The structure presented has been implemented in a prototype database, under an object-oriented database management system [POE99].

The big advantages of relational DBMS are: they have a solid theoretical foundation, they are stable and reliable after decades of debugging and engineering, and they are widespread.

In the end the choice of the DBMS is not the deciding factor for a successful database. It is whether we understand the relationships in the data correctly.

4.2.3 Visualisation

There are several ways to visualize the network data. They differ by showing more detail or more of the overall network:

1. Single entries are shown as a *table of fields and their attributes.* In hypertext format, all the fields referring to another entry can be made clickable, so that the user can navigate through the network by hand, brachiating from entry to entry. This is the most elementary way to visualize the database contents, and is employed for other kinds of databases, too. The path or network context is not visible in this view.

2. *Enhanced path layouts* have been employed for metabolic networks. To be able to build these, first some components in the database have to be defined as core components for linear paths. Then these linear paths can be laid out graphically in a way similar to textbook schemata [KP94a]. Cofactors and ubiquitous helper molecules like ATP — sometimes called external components — branch into and out of the main path, and feedback and feedforward loops for this path can be shown. The advantage is that the path looks similar to what biologists or biochemists are expecting it to look like. The disadvantage is that this expectation may be based on historic coincidence perpetuated by textbooks. For example it has been shown [KZL99] that textbook glycolysis, the classic example for metabolic paths, is only one of multiple possible paths. By assigning "core" status to selected nodes, a bias is introduced into the database. In this view, the visualisation context is limited to a linear path. The current implementation does not support this visualisation.

3. *Cascade trees* show the whole pathways reachable from a node. They have to break at loops to do so, and provide a middle ground between the linear path and the full network visualisation. No status assignment used only for the visualisation is needed. A disadvantage is that cascades can get long and hard to grasp when the subnet depicted increases in size. Their advantage over path layouts is their increased scope of context.

4. *Graph layout algorithms* are the most general visualisation of a network. Since optimal graph layout is an NP-hard problem [TBET98], generating the layout for large datasets has to be done with an heuristic algorithm. While the network does not look like a familiar cascade or path layout, graphs are able to show the whole network context. They are a good visualistation for queries of the type "what influences and is influenced by this molecule". Like trees and paths, they can be generated dynamically from the data.

5. *Hand-drawn maps* are crafted by human experts and can convey a high density of information [Tuf90], employing form and colour to show content. Since in highly connected networks it is difficult to show all interactions on a two-dimensional map, often a preselection for important compounds and interactions is made. Usually these maps are made clickable, so that the shapes representing the various molecules and reactions point to single entries in the database. Handcrafted maps have the advantage that they achieve a visual and layout quality that is hard to obtain by automated means. The disadvantages are that they have to be updated by hand if the database is updated, that they are static and can only show predefined pathways, and that they cannot be queried in turn with common computer programs. Maps are expressive in visualizing the whole network.

The current implementation of the interface is not limited to any one of those ways to show the data. Depending on his interest, the user can select the kind of display that best fits his needs, or switch between a high-level view for context and a detailed view to zoom into more information. No one single way to show the data is the best. A good combination of all ways is.

4.2.4 Class layout

Object-oriented methodology [Boo94] suggests an iterative development process. After specifying, a prototype implementation is created. The specification is modified and the prototype updated based on observations with this prototype, and with better understanding of the problem domain that comes with creating and using it.

This work lays out the conceptual design of a signal transduction database, and is the result of several such iterations. Implementing the design was extremely helpful in testing ideas and locating problem areas that I did not anticipate originally. Many of the insights about the database schema were gained by filling the database with increasingly detailed data, which then suggested improvements.

In this section I discuss reasons for some of the design choices and point out differences between the conceptual database and the current prototype implementation.

Molecule classification

A Molecule entry stands for every kind of agent that can signal. It may be beneficial to subclass Molecule with Basic, Family, Motif and State to enable type checking of the programming language for the different kinds of Molecule entries in the database. This is not currently implemented. A field named type in Molecule serves as a temporary fix. This classification contains different kinds of Molecule entries, not different kinds of molecules. A motif is not a molecule, and neither is a protein family. That is what sets it apart from the next kind of subclassification:

It is possible to subclass Molecule with classes like SmallMolecule, ProteinMolecule or EnzymeProteinMolecule. Some molecules share properties that set them apart from other kinds of molecules and that can be used to group them in such a class. For example, all Proteins have a sequence and all enzymes catalyze reactions.

There are efforts to build *Ontologies*[1] for biology in general and for biomolecules and genes in particular. Ontologies for biomolecule families resemble class hierarchies and usually subclass Molecule with classes like the ones discussed above.

One question is where to stop with this subclassification scheme. For enzymes there exists a hierarchical classification scheme, the EC nomenclature [otIUoBB92] which encompasses over 3.500 members.[2] A single entry of this, EC 2.7.1.112, represents all protein tyrosine kinases. Then the different protein-tyrosine kinases are in turn grouped into 23 families, which finally hold the proteins [Han94]. Sometimes, as in the case of the Ras superfamily, there are in turn subfamilies, which finally hold the proteins. A consequent use of sub-

[1] An Ontology is "the specification of a conceptualisation that is designed for reuse across multiple applications and implementations" [Kar00]. In other words it is a formal way to write down or store the concepts about the world or, usually, a certain aspect of the world, and their relations to each other.

[2] The EC nomenclature is based on the reactions that the enzymes catalyze, so it is actually a reaction classification. It is hierarchical. At the top are six broad classes of enzymatic activity; at the bottom are around 3,500 specific reaction types. EC classes are named with four numbers separated by periods.

classing must represent all those classes of molecules as classes of objects. The instances then would represent the actual molecules.

Clearly, for practical reasons it is not sensible to mirror the whole classification hierarchy in the class hierarchy. The classification of proteins tends to change over time, as new proteins are discovered and better understanding of the family relations is gained.

Since the best level of subclassification is not evident, the decision is made to start without any source-level subclassification of the Molecule class and introduce source-level subclasses only when practical experience indicates that it would be advantageous to do so.

Instead, a "soft" system of subclassification is used, which allows the database curator to introduce family relationships whenever they are needed, to arbitrary nesting levels. It is not constrained to a hierarchical relationship of single inheritance. Some proteins may be grouped as members of different strata. For example, a receptor Tyrosine kinase might be classified as a receptor as well as a Tyrosine kinase.

We rely on the domain knowledge of the annotators to uphold a correct classification structure. This assures correct biological relationships and relieves the burden on the input client developers. A good input client is as important as the relations in the database are, because no data can be entered effectively without it, and without data even the best database layout is worthless. Usually the supporting software takes more time to implement than the database layout.

Adding an Orthologues collection

It is an option to separate orthologues and paralogues and add a separate orthologues collection, in addition to the groups collection used for both at the moment. This information is already indirectly in the database, though, since orthologues have different species locations from each other. The common collection makes it clear that in both cases we have similar sequence or function, and we keep them together.

Adding a Connection class

Connection is not currently implemented in the database, but is in the process of being so. Connection does not inherit from Component, since it does not share Component's attributes. It is necessary to introduce this class, for the reasons given in 2.4.8. Connection is already part of the relational port of Transpath.

Reaction classification

By subclassing the generic Reaction class into Semantic and Mechanistic, we can employ type checking to differentiate the two kinds of reaction. Introducing a Translocation subclass for Mechanistic with a second location property allows us to represent translocations: the first location is where the molecule comes from, the second one, where it goes to. Since we again can use type checking to find translocations, this is faster and preferable to linking the Connections of the reaction with Locations, which is the alternative (compare 2.4.8).

To model a translocation in the current implementation, which does not have a Connection or a Translocation class, we have different state-entries for

the same molecule in the two locations, so that it can be seen by the reaction as two molecules, for example NF-κB(cyt) and NF-κB(nuc). This practice is a temporary fix, since it mixes the Molecule with the Location class and smears the definition of those two concepts.

Even further subclassing of the Mechanistic class in classes like Catalyzed and Complexation is possible. We could introduce one type of reaction per mechanism. The reasons for refraining from implementing this subclassification so far is that the data in the literature extracted does not mandate it: no detailed information about inhibition mechanisms is available.

Semantic and mechanistic reactions

Both kinds of reactions have their place in the database, semantic for overview information, mechanistic for detailed, primary information. The higher detail level makes the mechanistic view more usable for quantitative simulation.

The main weakness of the semantic view is the implicit assumption of only two states. Molecules with more than two relevant states cannot be represented correctly.

Detail level in modeling the reaction mechanism

When we use elementary primitives to represent complex concepts we obtain internal models where many primitives are needed to reflect one of the concepts. Complexity in concepts is reflected into complexity of structure. This reflection does not look similar to the concept and is difficult to construct by hand.

Complicated models of many primitives are only useful when they can be automatically generated from intelligible, conceptual models. The results obtained with such models have to be translated back into the language of the original concepts to be useful. An illustration: in programming, a few expressive high level language instructions are translated into many simple machine code instructions, which are totally unintelligible to most humans. This works because we have compilers, programs that translate our conceptual model that is intuitively understandable for humans into another one which is understood by the computer.

Applied to our particular problem this means that expressive reactions need fewer edges and nodes, since the edges are colored. Reactions can be modeled on different levels of abstraction:

On a low level is a *purely mechanistic* view, which consists of elementary reactions only (see 3.3.2). Nothing has to be known about any of the molecules or reactions involved, and reactions are, with rare exceptions, limited to aggregation of two molecules, transformation of a complex into another, and dissociation of a complex into two molecules. This leads to "dumb" reactions and many intermediary species, or nodes.

On a high level, we model a complicated regulatory process as a single Reaction entry, which allows us to quantify relations between enzyme, cofactors, activators and inhibitors in the reaction description. These species may compete and the state of the enzyme in the reaction may depend nonlinearily on the relative concentrations of activators and inhibitors. One approach to capturing this kind of competition can be seen in the elaborate schemes of enzyme kinetic formulas[Seg75]. The whole network of elemental reactions between all these

molecules is integrated into one equation. This leads to "smart" reactions, and only one node per reaction.

The mechanistic reactions we use represent a middle ground between the two extremes. We break up complicated regulatory mechanisms into smaller steps, but integrate the enzymatic catalysis into a single Reaction. If we chose to forget about the meaning of the enzyme we would need more nodes to model the mechanism of the process — compare 3.8 and 3.6.

For a signaling reaction in the cytosol, we usually have two or three signaling inputs to the reaction, either two molecules that interact directly, or one or two molecules aided by an enzyme.

The decision as to which molecules will react in a real-life situation will depend on the concentrations of the molecules, on their condition, and on the binding constants. Thus competition or inhibition decisions based on the mechanistic model can only be seen during actual simulation of the mechanism. To support this we need contextual information for the molecules and reactions, like stoichiometric indices and speed constants.

Since an enzyme might work on different substrates with different speeds, we cannot store these parameters in the Molecule entry and use the Reaction for this purpose.

4.2.5 Interface

There are two common ways to make biological databases available:

1. World wide web pages, often generated dynamically by server side scripts
2. FTP-download of the interface software and data flatfiles, for local installation

When considering which kind of interface to develop for querying and filling the database target properties are: portability, ease of use, support for multiple concurrent users, and accessibility.

Web interface

The web-based approach has several advantages: first the user does not need to download and install the software. Since there is no local installation, portability is not an issue, the database will be available to anyone with a web browser. Second, web pages have a broad user community and the metaphors they use, links and forms, are well understood and therefore easy to use. Third, multiple users are supported automatically through the web server. Fourth and last, the user always and from anywhere in the world has access to the latest version of the software and the latest release of the database without having to update explicitly. Hence a WWW interface is chosen as the primary interface for the database.

Java (see Appendix A) is a portable, object-oriented language that can be used flexibly for web applications, as applets on the client side, as servlets on the server side or for stand-alone applications. It is supported by several object-oriented database management systems. These two properties make it the language to fit both the object-oriented modeling of the database and the web-based publication of its contents. Therefore Java was chosen as the implementation language.

General architecture

There are several designs to implement the web interface:

1. Straight pregenerated HTML pages that hyperlink to each other.
2. Applets that retrieve data from the database over the net.
3. Template HTML files that are dynamically filled with data from the database, like Java Server Pages.[3]
4. Servlets that dynamically generate HTML pages filled with data from the database.

Static HTML pages are technically the simplest solution. They have to be re-generated every time the database content changes. Their disadvantage is the missing computational capability, which makes it impossible, for example, to answer pathway queries. Applets have the disadvantage that they have to send requests for data over the network, which can be slow if data is retrieved in small quantities. Template imbedded Java code is a good solution if there are considerable amounts of boilerplate HTML, and little snippets of code. If the pages mainly consist of dynamically generated content, it is more sensible to embed the HTML into the Java code using normal servlets.

A web-based input client makes it easily possible to update the database via the network. There is no limit to where in the world the annotator sits, and multiple annotators can be supported.

4.3 Previous work

This section gives an overview on what databases, models and simulation engines for bionetworks exist.

4.3.1 Databases

Databases are static models of networks. The static model contains the signaling compounds and their interactions. The interactions include additional information like parametric constants.

Several other efforts to manage signal transduction data exist:

CSNDB (`http://geo.nihs.go.jp/csndb.html`) The cell signaling networks database [TIK00] strives to be a general database on signal transduction, and is the only other database with comparable aims. CSNDB is developed and curated by Takako Takai-Igarashi, National Institute of Health Sciences, Japan. CSNDB provides a rule-based linear pathway visualisation in addition to plain entries. In contrast to Transpath, CSNDB focuses on semantic reactions. Information is extracted mainly from the review literature. With friendly consent of Dr. Takai-Igarashi, the data set of CSNDB is available under the Transpath interface (compare 2.5.6).

[3] Java Server Pages, JSPs, are special kinds of servlets that allow to embed Java source code into HTML pages, to be compiled and evaluated on the server side when the pages are requested.

KEGG (`http://www.genome.ad.jp/kegg/kegg.html`) The Kyoto Encyclopedia of Genes and Genomes [KG00] is developed at the Institute for Chemical Research, Kyoto University under Minoru Kanehisa. KEGG focuses on genes and metabolic pathways and is based on binary relations. It has an additional database module, PATHWAY, to store information about signal transduction intercations, in the semantic view. KEGG provides hand-drawn maps as an additional visualisation.

stke (`http://www.stke.org/`) The signal transduction knowldedge environment is developed by Stanford University Libraries in collaboration with Science. It is a non-formalized collection of information on signal transduction in form of an online publication, consisting mostly of review articles and clickable maps.

SPAD (`http://www.grt.kyushu-u.ac.jp/spad`) The The Signaling Pathway Database is maintained by Hakozaki Higashi-ku, Japan Graduate School of Genetic Resources Technology, Kyushu University. It provides hand-drawn clickable maps as the only interface, categorized by extracellular signaling molecule.

None of these databases stores the states of the signaling molecules. For many cases, molecular states are necessary to describe adequately the signaling process. Consequently, all of these databases strive to be information sources only, and do not support simulation needs. Most of the databases only support direct edges between molecules, corresponding to a common graph, and do not model the reactions as separate nodes (CSNDB is the notable exception).

Very Large System Integration (VLSI) Design

VLSI design [Ger99] is used for the modeling of electronic signaling networks or circuits. The similarity between biological signaling networks and microchip signaling layouts is suggestive [HHLM99] and the assumptions about modularization (see 4.6) suggest application of the same techniques that are used in electronic VLSI design for modeling the very large biological systems. VLSI modeling is conducted on multiple layers of abstraction, and one of the difficulties is to find analogues for biological systems. In that context the gate-level logic models seem to be most amenable.

McAdams [MS95] used this approach by modeling gene-regulatory networks, mixing in nonlinear differential equations for selected critical control elements, and Arkin [AR94] and coworkers are developing an environment dubbed Bio/SPICE based on these assumptions. Any such application will ultimately need a structured "parts catalogue" or library as is provided by the database developed in this thesis.

4.3.2 Simulations

Simulations are dynamic models of networks. The starting conditions are the input to the model.

Network models of all kinds have been used on research topics as diverse as electrical circuitry, traffic control and gene-regulatory or metabolic networks. Since we are interested in ways to model networks, we present an overview based

on the kind of model used, not on the kind of application domain, but will cite examples from biological modeling. The classification of methods follows [HT98].

Logical models

These are the simplest possible models. Components in the model are in one of two states, "on" or "off".

Boolean Networks A Boolean network calculates the state of a node in the network from the state of incoming connected nodes. They are able to express positive and negative feedback loops. Parameters are a set of rules and a set of Boolean tokens that defines which cells are on in each iteration of the network. It is computationally cheap and has been used before to model gene-regulatory networks [TR99]. Since the model is Boolean only, no quantities can be expressed. Thus amplification, delays and thresholds cannot be modeled intuitively.

Discrete models

These models do not use exact concentrations or rate constants. They represent a middle ground between the more exhaustive differential equation systems and the simplified logical models.

Petrinets Petrinets have a mature theoretical background [Rei91], and numerous tools for analysis and running simulations exist. There are a lot of variants, including colored and timed Petrinets, but unfortunately, as these variants get more expressive, they are more difficult to treat with the basic theory. Interesting system properties that can be analyzed with Petrinets are liveness and reachability. Hofestädt and Thelen [HT98] used Petrinets to simulate metabolic networks.

Stochastic Activity Nets (SANs) SANs are a variant of Petrinets, which allow additional conditions upon which transactions fire, thereby making it easier to model behavior without introducing many additional places. Switching time here is based on probability distributions, overcoming some limitations most other discrete models have. These models were investigated further and are discussed in more detail in 3.3.

There are still only discrete sets of tokens moved in the network. One can assimilate floating-point behavior by large numbers of tokens, so that one token in effect equals the smallest possible fraction of a unit to be represented. Mounts and Liebman [ML97] used a SAN model for blood coagulation.

Rule-based systems Implementations include Prolog, Lisp, or expert system shells. The network is modeled as a collection of IF x THEN y rules. A rule fires, when the expression in its preconditions x holds, setting the values in its post-conditions y. There are efficient algorithms, like RETE,[For82] to re-evaluate if a rule fires when some arbitrary condition of the network is changed. The range for parameters can go from Boolean markings up to floating point numbers, depending how the rules are formulated. Rules

represent interactions. There are commercial tools to do the computation. The big advantage is that this model closely mimics the way biologists talk about signal transduction, and that the rules for the interactions can be arbitrarily complex, but will also work with crude information. Brutlag [BGM91] used rules to model enzyme behavior (for DNA polymerase).

Neural Networks/Parallel Distributed Process-networks A layered network of neural units with input layer, hidden inner layers, output layer plus training program for weights is the neural network approach. This method goes in the direction of machine learning. Only connectivity and experimental data are needed. The disadvantages are that extensive datasets are needed for training, and, while the model may be able to predict the outcome, it *explains* nothing. Bray [Bra90] had an example application.

Continuous models

These are the most detailed models and they are commonly used for quantitative simulation. They need kinetic parameters and have to date been used most often. Because of their hunger for information usually only small, well known systems can be modeled.

Quantitative modeling gives the highest predictive precision and allows the quantification of signal transfer.

A mature theory and several databases exist for metabolic networks and their analysis, which have been largely modeled with differential equations [SDF99, KP94b]. In metabolic networks the model is that of small molecules being transformed by enzymatically catalyzed reactions. Thus there is always a flow of mass through the network. Nothing is said about the regulation of the enzymes - if they are present, they are assumed to work. In regulatory networks, information is flowing, and often does so without mass being passed on. Consider the catalysis in regulatory cascades: here you have futile cycles between the active and inactive state of the enzymes from a mass-conservation point of view, but what really counts is the passing of information that manifests in this state-change. These processes will not yield usable results with present day tools for metabolic network analysis.

Differential equations The network is modeled as a collection of differential equations, where reaction rates and compound concentrations are the variables. These are then solved numerically for each time step. Stoichiometric coefficients, starting concentrations and rate constants for all interactions are needed [MS95, S$^+$99].

Stochastic Processes An alternative to differential equations, which uses fewer assumptions about the distribution of molecules in the cell and is advantageous for low molecule copy numbers, is the use of stochastic processes [Gil77]. In this case the absolute molecule numbers have to be estimated, and the system behaviour is predicted by sampling multiple runs and averaging over them.

Colored Petri Nets Some colored Petri nets allow arbitrary attributes on their markings. Küffner et al. [KZL99] constructed a Petrinet which used this feature to change the marking attributes (representing concentrations

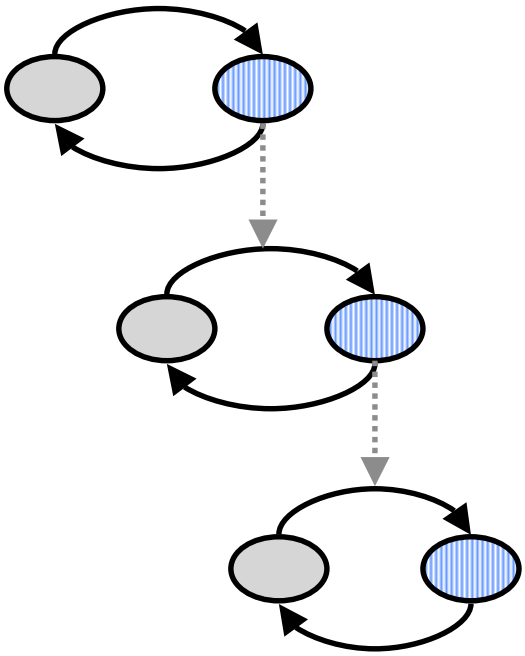

Figure 4.1: Information flow in signaling cascades. Solid ellipses are inactive enzymes, striped ones active enzymes. Solid arrows indicate flux of mass while dashed arrows indicate flow of information.

of compounds) instead of the number of markings. All the comments made for differential equations apply here.

4.4 Applications

Everything that has been said in 4.2.4 for static models applies for dynamic models, too: *The best representation for understanding the networks will again be one that depicts the modeled entities and their reactions similarly to the way we think about them.*

All simulations and analyses with the database can yield no other result than what the database provided. With the data still being fragmentary, results of such analyses may vary significantly from the real outcomes in the laboratory. As more and more data is added to the database and knowledge grows, the model will come closer to the real biological system, and so will the predictions. The methods that have been employed in this work and the tools developed to do so, can be applied to any later, more comprehensive dataset.

Often only the general nature of molecular interaction is known. Stoichiometric factors, rate constants and reaction rates are missing in many cases. Since timing can have drastic effects on signal network behavior [Ger99] it may not be possible to conduct detailed simulations before such data becomes available.

4.4.1 Path alignment

Our main assumption was that only the shortest paths are relevant. There are some situations where a longer path makes sense, for example in a cascade which serves to amplify the signal. We ignored these, to get something we could work with.

Hypotheses

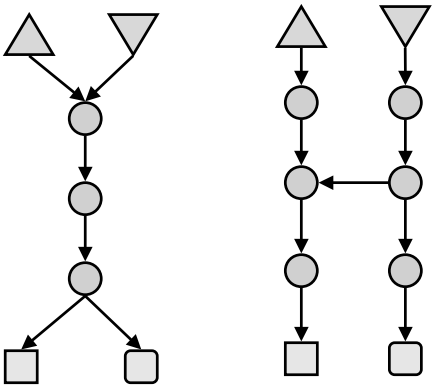

Figure 4.2: Pathway hypotheses. Left, generic pathways link to a variety of receptors and transcription factors. Right, different classes of receptors and factors have their own specific pathways, which can crosstalk.

One hypothetical model for signal transduction is that a set of receptors leads to a few general pathways, which funnel the signal to a set of transcription factors. Specificity is mainly achieved by differential expression of receptors and factors between different cell types.

An alternative hypothetical model is that different kinds of receptors connect to different, specific signaling pathways, and these pathways may interact via crosslinking molecules.

Fig. 4.2 shows the hypotheses graphically.

Interpretation of the results

If the first hypothesis is true, the paths should be similar to each other in the middle, since the middle is the part of the sequence that reflects the transduction channel.

If, on the other hand, the alternate hypothesis is true, then the the paths should be similar to each other at their ends. The discussion refers to the similarity tree in Fig. 3.2.

Similar paths will cluster together. If we find that in one branch of the tree all the paths clustered together vary in both their signaling molecule and their transcription factor, then the clustering is due to a shared inner stretch of sequence, and the funneling model is true for this group. For example, the Ras-MAP kinase cascade connects EGF, PDGF, ACTH and CRH to AP-1, which means to Jun and Fos, Elk-1 and CREB2.

If we find that all paths in one branch of the tree start from the same signaling molecule or family of signaling molecules, then the clustering is due to a signaling pathway that is specific for that family. This pathway is connected to other pathways by means of a crosstalk point. This is for example the case for the Interleukines, which signal over STAT proteins. The same reasoning applies to clusters ending in a common transcription factor or family of factors.

One caveat is that the quality of the alignments depends strongly on the detail and quality of the pathway descriptions in the database. One must use the same level of detail throughout the database. If we use many reactions to describe one step in detail, this step will contribute to the alignment more than it should. If we use one reaction to subsume a whole signaling cascade then the cascade will not contribute as much as it should to the alignment. For the example pathways mentioned, two such cases in the CSNDB data can be found. The "CRH → ACTH" reaction describes extracellular hormonal signaling instead of signal transduction signaling and the "MAPK cascade" molecule entry represents all the reaction steps and molecules of the MAP kinase cascade.

The algorithm and method produces alignments that grow more useful as the number and quality of pathways in the database grows.

If there are mixtures of both modes, where a specific pathway is hooked up with a common pathway, the length of the string for each sub-alignment decides the outcome of the clustering. One example is the IL-9 STAT signaling pathway, which leads into the MAP kinase cascade.

Conclusion

The tree shows that the data suggest neither of the two extreme hypotheses is correct, and that a mixed model would be more appropriate.

4.4.2 Boolean networks

The inference process behind Booelan networks closely mimics the way biologists are often forced to think about signaling processes: one active species goes up in concentration, suppressing another one, which goes down or activates another one, which also goes up. Even with this over-simplifying approach, networks rapidly grow unpredictable when evaluated manually.

The Boolean approach formalizes this process and makes it possible to predict such outcomes for large networks. It finds emergent behavior which is not obvious from inspection of the network, like the oscillatory cycle in our example in Fig. 3.4.

It has been discussed that it is possible to capture oscillatory behavior with Boolean networks for regulatory networks that eludes continuous models of the same system [HL99]. We show that it is easy to evoke such behavior within Boolean networks applied to signal transduction cascades.

The model as it stands predicts the persistent activation of the feedback cycle. Even after withdrawal of the stimulus later on, the fully activated cycle stays active, as shown in basin number three. Quantitative simulations [BI99] suggest that this is only the case for strong and sustained activation. The Boolean network is able to capture the qualitative behaviour, but not the quantitative aspects.

Such quantitative models are the subject of the next section. For now we note that Boolean networks provide a conceptual, rough estimate how a signaling system might behave.

4.4.3 Petri nets

We used Petri nets to conduct a quantitative simulation, which depends on numeric paramters for the reactions. If as a crude simplification k_f, k_b and k_p are set to one, the dynamic behavior totally breaks down. This is to be expected: equilibrium or catalyzed forward and backward reactions progress with the same speed, and consquently token numbers oscillate around their starting values, varying only according to the probability distribution. It shows that rates are critical for the behavior of the system.

Stiffness

Straightforward Petri net simulations of chemical kinetic systems are *stiff* [AS92]. The fastest process in the whole system will dictate the stepwidth of the simulation, even if the interesting changes are taking place at a much slower rate and need a bigger stepwidth.

For example, a simple equilibrium reaction after a short while will reach a state where the markings oscillate minimally around the equilibrium value. If we treat the forward and the backward reaction independently and their rates are a thousand times as big as the rate of the reaction of interest, we will produce two thousand superfluous steps for each step we are interested in. The amount of useless data we generate is uneconomical and may even be prohibitive.

The stiffness makes automated generation of Petri net models from the database problematic, and proves limiting for the size of the simulated network. In the original model several other subnets were included, for example a subnet for the activation of PKC, which contained fast equilibrium reactions only and made the size of the simulation output too big to handle.

In the case of PKC, running the PKC subnet alone showed that after a second an equilibrium of 20 nM is reached. Since this process is much faster than the cascade process itself, the concentration of PKC could then be given as constant 20 nM for the cascade model. In other cases, such simplifications may not be possible, especially when feedback processes influence the equilibrium level later in the simulation.

Therefore, to make qualitative analyses with Petri nets feasible, analytical methods to simplify the model would have to be employed. One possibility of course is to have a human examine the networks and conduct such analyses. For automated prediction of network behavior, however, they also would have to be automated.

For the modeling approach pursued here, Petri nets provide no advantage over systems of differential equations as far as parameter sensitivity is concerned.

The *telescopic* property of Petri nets, which means that nets may be composed from subnets, is valuable for successful model building by humans as it helps to keep the complexity at a manageable level.

Even more important is the graphical map that the Petri net directly furnishes for the network of interactions. It helps to localize interaction parameters

and understand the relation of elements to each other. For a differential equation model that was built for comparison purposes, the Petri net maps proved indispensable.

4.5 Status and future of the database

The database is available to the academic community on the WWW under `http://transpath.gbf.de/`.

Biobase GmbH, a company specializing in databases for gene-regulation, has aquired the rights for Transpath. The implemented prototype is used by Biobase GmbH for storage and annotation of signaling pathways. Currently five annotators are entering data into the system. 2373 Molecule are linked by 2684 Reaction entries, for which 405 References have been evaluated. Biobase is further developing the database.

Many companies have a huge investment in know-how and licences in relational databases. They need a good reason before they accept the cost to retrain their staff and convert their legacy data — or worse, administer and curate both systems concurrently. Relational databases work reliably. Object-oriented DBMS are younger, and less mature and reliable, which is a drawback if a company depends on the database's availability. These are some of the reasons why OODBMS have a market share below 10 percent, in spite of being theoretically superior.

It is easy to port tables from several relational databases into one and link them to integrate different databases. Biobase offers other databases, which reside in a relational DBMS. To normalize data between these databases, they share certain tables, for example the Reference table for literature citations.

To integrate Transpath with their other databases, Biobase GmbH is creating a relational version of Transpath. This version will be sharing the References of the common Reference table, and provide a framework for gene-regulatory networks between the transcription factors of the TRANSFAC database.

Cytomer^{TM}[W^{+}00] is a relational database that specializes in representing hierarchically organized expression spaces like organisms, systems, organs, tissues, developmental stages and cell types, unified in a hub table. In the relational version of Transpath, this table will replace the corresponding fields from location.

4.6 Open questions

It is an old adage that biologists know more about biology then they are able to express. There are several interesting assumptions and questions about signal transduction networks which have not been investigated systematically, as such an analysis is not possible without an extensive data collection:

What is a signal transduction pathway?

Even though this whole work is preoccupied with signal transduction or signaling pathways for short, there is no common definition what a signal transduction pathway *is*.

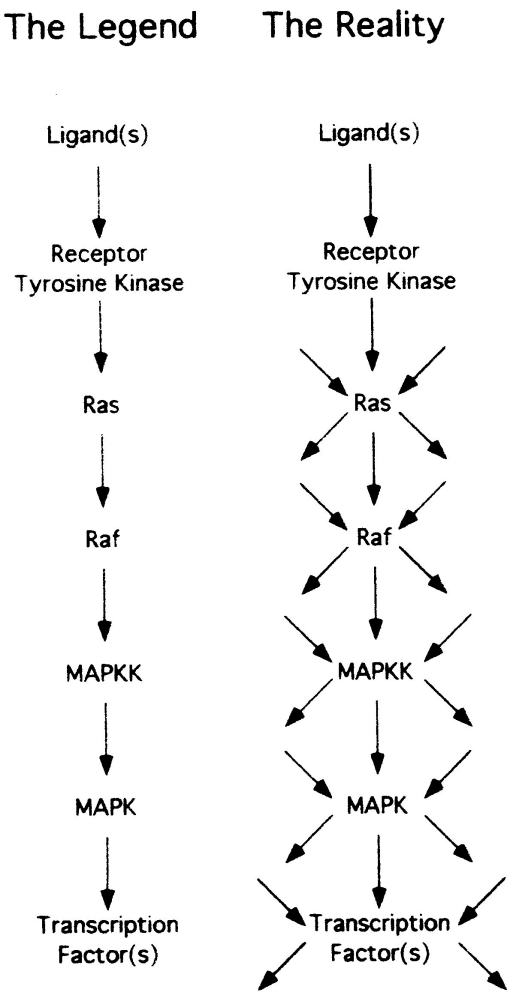

Figure 4.3: Ras pathway crosstalking. Image taken from an article in Cell[ME98] discussing the nature of pathways.

In biological papers, pathways are named after key molecules ("MAPK cascade"). In that context, pathway refers to a more or less linear path from receptor to transcription factor, even though usually some of the molecules involved are also found in other pathways. The influence that one pathway exerts on another one via these molecules is called crosstalk. The question is how to define a boundary for a pathway under these circumstances. The classification of molecules into pathways has historical reasons and is not explicitly based on qualitative differences of the interactions. One reason for this is that it is easier to think about the network as pathways connected by crosstalks.

It is not known to which extent crosstalking influences the signaling behavior. Are crosstalks mere by-products of a system that is mainly linear in its transduction activities from plasma membrane to nucleus, or are they integral to the system?

While detailed data may be available for single elements in the network, no such data exists that measures the quantitative properties of pathways, like their length, size, number of feedback cycles, number of crosstalks between pathways and area reachable from any point in the network. Probably this is so because we have no generally accepted definition as to what constitutes a pathway in the first place.

Is the network built from modules?

Man-made complex systems such as airplanes, microprocessors or software applications share one common trait: they are built up from modules.

The more interconnections exist between the different parts of a system, the harder it gets to predict how the system will react. It also gets harder to change any part of the system without upsetting another part. When systems reach a certain size, they will become unmanageable and non-understandable without decomposition into modules.

If we compose the overall system of simpler subsystems or modules, we can divide and conquer[4]. Each module must be simple, so that we can understand and predict the inner behavior. Then we forget about the inner workings of the module and regard its overall function. We have created a building block, decoupled from the internal interaction network. We now can construct a system by plugging together modules and, again, we can understand the workings of this higher-level system and are able to predict its behavior. In turn we can use it as a module for building an even higher-level system.

Because this method of recursive modularization is the only one known to us which makes construction of large complex systems possible, it is tempting to suppose that biological systems work the same way. And on higher spatial levels of the organism it is evident that this is the case (see 1.2.1). But it is not clear how far this holds for intracellular signaling networks, even though we would expect to see modularization for a system with hundreds of components.

It would certainly facilitate understanding of the signaling network if we could come up with a model for modules. The question is what criteria should be used to define the boundaries between these modules? Finding a correct segmentation of the signaling network is the most pressing challenge that lies ahead in this area.

4.7 Summary

Transpath is an information system on signal-transduction networks. It focuses on pathways involved in the regulation of transcription factors. Molecules and reactions are the nodes in a signaling graph. They are stored in an object-oriented database, together with information about their location, quality, family relationships and signaling motifs. Also stored are links to other databases and references to the original literature. Transpath differentiates between the

[4]There are two approaches in building large systems, the top-down and the bottom-up approach. In the top-down approach we start from the overall system and seek to segment it into simpler subsystems. In the bottom-up approach we start from simplest subsystems and build them up to the overall system. In practice, usually a mix of both methods is employed, since both are expressions of the same idea: namely, that the system is composed from simpler subsystems.

states of a signal molecule, and can adequately describe the reaction mechanisms of signaling interactions.

It is available over the web through a Servlet-based interface, that creates dynamic content directly from the contents of the database. Pathway query mechanisms and several kinds of display are provided for the database in addition to text-based queries and information on single entries.

We show that the database is useful for analysis of the signaling network and can provide the basis for simulations.

"*A problem worthy of attack*
proves its worth by fighting back."
— Paul Erdős (1913–1996)

Bibliography

[ABD+95] Malcolm Atkinson, François Bancilhon, David DeWitt, Klaus Dittrich, David Maier, and Stanley Zdonik. The object-oriented database system manifesto. `http://www.cs.cmu.edu/People/clamen/OODBMS/Manifesto/htManifesto/Manife%sto.html`, 1995.

[ABS+97] H. Aberle, A. Bauer, J. Stappert, A. Kispert, and R. Kemler. β-catenin is a target for the ubiquitin proteasome pathway. *EMBO J.*, 16:3797–3804, 1997.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, 25:3389–3402, 1997.

[Apw00] R. Apweiler. *CCP11 Newsletter*, 10, 2000.

[AR94] Adam Arkin and John Ross. Computational functions in biochemical reaction networks. 67:560–578, 1994.

[AS92] Robert A. Alberty and Robert J. Silbey. *Physical Chemistry*. John Wiley & Sons, 1992. ISBN: 0–471–62181–1.

[BA00] Amos Bairoch and Rolf Apweiler. The SWISS-PROT protein sequence database and its supplement trEMBL in 2000. *Nucleic Acids Res.*, 28:45–48, 2000.

[Bar98] Douglas K. Barry. Object-oriented databases. `http://www.odbmsfacts.com/`, 1998.

[BBD+00] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. Sonnhammer. The pfam protein families database. *Nucleic Acids Res.*, 28:263–266, 2000. Based on human crafted multiple alignments and HMMs.

[BGM91] D. L. Brutlag, A. R. Galper, and D. H. Millis. Knowledge-based simulation of DNA metabolism: prediction of enzyme action. *CABIOS*, 7(1):9–19, 1991.

[BI99] U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *Science*, 283:381–387, 1999.

[Boo94] Grady Booch. *Object-Oriented Analysis and design*. Addison–Wesley, 1994. ISBN: 0–805–35340–2.

[Bra90] D. Bray. Intracellular signalling as a parallel distributed process. *J. Theor. Biol.*, 143(2):215–31, 1990.

[Ce97] R. G. G. Catell and Douglas K. Barry (editors). *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997. ISBN: 1–558–60463–4.

[CGK99] F. Corpet, J. Gouzy, and D. Kahn. Recent improvements of the prodom database of protein domain families. *Nucleic Acids Res.*, 27(1):263–267, 1999. Automatically generated based on PSI-BLAST searching.

[Cha98] Akmal B. Chaudhri. ODBMS ressources. `http://www.soi.city.ac.uk/~akmal/html.dir/resources.html`, 1998.

[Chr00] Maik Christensen. Datenintegration biologischer Datenbanken. Master's thesis, Technische Universität Braunschweig, 2000.

[CRB95] G. B. Cohen, R. Ren, and D. Baltimore. Modular binding domains in signal transduction proteins. *Cell*, 80:237–248, 1995.

[DM91] Richard Durbin and Jean Thierry Mieg. A C. elegans database. documentation, code and data available from anonymous FTP servers at lirmm.lirmm.fr, cele.mrc-lmb.cam.ac.uk and ncbi.nlm.nih.gov, 1991.

[EW93] E. S. Egan and R. A. Weinberg. The pathway to signal achievement. *Nature*, 365:781–83, 1993.

[Fey67] Richard Feynman. *The character of physical law*. MIT Press, 1967. ISBN: 0–262–56003–8.

[For82] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[Ger99] Sabih H. Gerez. *Algorithms for VLSI Design Automation*. Addison–Wesley, 1999. ISBN: 0–471–98489–2.

[GHuJV86] Erich Gamma, Richard Helm, and Ralph Johnson und John Vlissides. *Design Patterns: Elements of reusable object-oriented software*. Addison–Wesley, 1986. ISBN: 0–201–63361–2.

[Gil77] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[Gra00] Niels Grabe. Alibaba2: Context specific identification of transcription factor binding sites. *In Silico Biol.*, 1:0019, 2000.

[Gro00] Thorsten Gross. Erweiterung des java-clients für die relationale datenbank TRANSFAC um die objektorientierte datenbank TRANSPATH. Master's thesis, Technische Universität Braunschweig, 2000.

[Hal30] J. B. S. Haldane. *Enzymes*. MIT Press, 1965 (reprint of original from 1930).

[Han94] Steven K. Hanks. Protein kinase classification. `http://www.sdsc.edu/kinases/pkr/pk_catalytic/pk_hanks_class.html`, 1994.

[HBFB99] K. Hofmann, P. Bucher, L. Falquet, and A. Bairoch. The PROSITE database, its status in 1999. *Nucleic Acids Res.*, 27:215–219, 1999. Based around regular expression patterns and profiles.

[Hei97] Nick Heinle. *Designing with JavaScript: Creating Dynamic Web Pages*. O'Reilly, 1997. ISBN: 1–56592–300–6.

[Heu92] Andreas Heuer. *Objektorientierte Datenbanken*. Addison–Wesley, 1992. ISBN: 3–89319–315–4.

[HHLM99] H. L. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(SUPP):C47–C52, 1999.

[HHP99] J. G. Henikoff, S. Henikoff, and S. Pietrokovski. New features of the blocks database servers. *Nucleic Acids Res.*, 27:226–228, 1999. Based around automatic ungapped alignments.

[HL99] Vassily Hatzimanikatis and Kevin H. Lee. Dynamical analysis of gene networks requires both mRNA and protein expression information. *Metabolic Engineering*, 1:275–281, 1999.

[HT95] C. S. Hill and R. Treisman. Transcriptional regulation by extracellular signals: mechanisms and specificity. *Cell*, 80:199–211, 1995.

[HT98] R. Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. *In Silico Biol.*, 1:0005, 1998.

[Huf00] Otter. `http://www.caida.org/tools/visualization/otter/`, 2000. Otter is a historical CAIDA tool used for visualizing arbitrary network data that can be expressed as a set of nodes, links or paths. Otter was developed to handle visualization tasks for a wide variety of Internet data, including data sets on topology, workload, performance, and routing. Otter is in maintenance rather than development mode.

[Hun00] Tony Hunter. Signaling — 2000 and beyond. *Cell*, 100:113–127, 2000.

[HW00] D. Hanahan and R. A. Weinberg. The hallmarks of cancer. *Cell.*, 100(1):57–70 Review, 2000.

[Jav00] *JavaScript Documentation and ECMA–262 JavaScript Language Specification*, 2000.

[Kar00] Peter D. Karp. An ontology for biological function based on molecular interactions. *Bioinformatics*, 16:269–285, 2000.

[Kau93] Stuart A. Kaufmann. *The Origins of Order: Self Organisation and Selection in Evolution*. Oxford University Press, 1993. ISBN: 0–19–507951–5.

[KE99] A. Kemper and A. Eickler. *Datenbanksysteme*. Oldenburg, 1999. ISBN: 3–468–25053–1.

[KG00] M. Kanehisa and S. Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 28:29–34, 2000.

[KHWB97] B. N. Kholodenko, J. B. Hoek, H. V. Westerhoff, and G. C. Brown. Quantification of information transfer via cellular signal transduction pathways. *FEBS Letters*, 414:430–434, 1997.

[Knu97] Donald E. Knuth. *The Art of Computer Programming : Fundamental Algorithms (Vol 1, 3rd Ed)*. Addison–Wesley, 1997. ISBN: 0–201–89683–4.

[KP94a] P. D. Karp and S. M. Paley. Automated drawing of metabolic pathways. 1994.

[KP94b] P. D. Karp and S. M. Paley. Representations of metabolic knowledge: Pathways. 1994.

[Kra97] Gerhard Krauss. *Biochemie der Regulation und Signaltransduktion: Das moderne Lehrbuch für Chemiker, Biochemiker, Biologen, Mediziner*. VCH, Weinheim, 1997. ISBN: 3–527–29393–0.

[KZL99] Robert Küffner, Ralf Zimmer, and Thomas Lengauer. Pathway analysis in metabolic databases via differential metabolic display. *GCB*, 1999.

[LFSSC93] B. M. Laoide, N. S. Foulkes, F. Schlotter, and P. Sassone-Corsi. The functional versatility of crem is determined by ist modular structure. *EMBO J*, 12(3):1179–1191, 1993.

[MA97] Harley H. McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *Proc. Natl. Acad. Sci. USA*, 94:814–819, 1997.

[ME98] P. Meier and G. Evan. Dying like flies. *Cell*, 95:295–298, 1998.

[ML97] W. M. Mounts and M. N. Liebman. Qualitative modeling of normal blood coagulation and its pathological states using stochastic activity networks. *Int. J. Biol. Macromol.*, 20(4):265–281, 1997.

[Mon97] M. Montminy. Transcriptional regulation by cyclic AMP. *Annu. Rev. Biochem.*, 66:807–22, 1997.

[MS93] J. F. Meyer and W. H. Sanders. Specification and construction of performability models. In *Proceedings of the Second International Workshop on Performability Modeling of Computer and Communication Systems*, 1993.

[MS95] H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–56, 1995.

[OG98] L. A. O'Neill and C. Greene. Signal transduction pathways activated by the IL-1 receptor family: ancient signaling machinery in mammals, insects, and plants. *J Leukoc Biol.*, 63(6):650–7 Review, 1998.

[otIUoBB92] Nomenclature Committee of the International Union of Biochemistry and Molecular Biology. *Enzyme Nomenclature*. Academic Press, 1992. ISBN: 0–122–27164–5.

[P+00] Steven Pemberton et al. XHTML™1.0: The extensible hypertext markup language. `http://www.w3.org/TR/xhtml1/`, 2000.

[Pag96] R. D. M. Page. TREEVIEW: An application to display phylogenetic trees on personal computers. *Computer Applications in the Biosciences*, 12:357–358, 1996.

[POE99] *POET Object server & Java SDK*, 5.1 edition, 1999.

[Rag00] Dave Raggett. HTML tidy — HTML parser and pretty printer. `http://www.w3.org/People/Raggett/tidy/`, 2000.

[RCPÖ99] I. Rojdestvenski, M. Cottam, Y.-I. Park, and G. Öquist. Robustness and time-scale hierarchy in biological systems. *BioSystems*, 50:71–82, 1999.

[Rei91] Wolfgang Reisig. *Petrinetze: eine Einführung*. 1991. ISBN: 3–540–16622–X.

[S+] Alexander Spirov et al. Deciphering the genetic networks: inferring gene relationships from expression data for drosophila segmentation genes. `http://academic.mssm.edu/molbio/tmp/`.

[S+99] E. Selkov et al. Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, 15(9):749–758, 1999.

[SB97] Ilana Stancovski and David Baltimore. NF-κB activation: The IκB kinase revealed? *Cell*, 91:299–302, 1997.

[SC97] Randal L. Schwartz and Tom Christiansen. *Learning Perl, 2nd. Ed.* O'Reilly, 1997. ISBN: 1–56592–284–0.

[Sch98] Manfred Schneider. Links on objects & components. `http://www.cetus-links.org/`, 1998.

[SDF99] S. Schuster, T. Dandekar, and A. D. Fell. Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *TIBTECH*, 17:53–60, 1999.

[SDG96] R. K. Sunahara, C. W. Dessauer, and A. G. Gilman. Complexity and diversity of mammalian adenylyl cyclases. *Annu Rev Pharmacol Toxicol*, 36:461–480, 1996.

[Sed92] Robert Sedgewick. *Algorithms in C*. Addison–Wesley, 1992. ISBN: 0–201–51425–7.

[Seg75] Irwin H. Segel. *Enzyme Kinetics*. John Wiley & Sons, 1975. ISBN: 0–471–77425–1.

[SQL97] *ISO/IEC 9075:1992 Information technology – Database languages – SQL*, 3 edition, 1997.

[SW63] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. Univ of Illinois Pr., 1963. ISBN: 0–252–72548–4.

[T+99] C. M. Thomas et al. *FEBS Letters*, 458:247–251, 1999.

[TBET98] Ioannis G. Tollis, Giuseppe Di Battista, Peter Eades (Editor), and Roberto Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998. ISBN: 0–133–01615–3.

[THG94] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[TIK00] Takako Takai-Igarashi and Tsuguchika Kaminuma. A pathway finding system for the cell signaling networks database. *In Silico Biol.*, 1:980112, 2000.

[TR99] D. Thieffry and D. Romero. The modularity of biological regulatory networks. *BioSystems*, 50:49–59, 1999.

[Tuf90] Edward R. Tufte. *Envisioning Information*. Graphics Press, 1990. ISBN: 0–961–39211–8.

[V+01] J. Craig Venter et al. The seqence of the human genome. *Science*, 291:1304–1351, 2001.

[Voi00] Eberhard O. Voit. *Computational Analysis of Biochemical Systems*. Cambridge University Press, 2000. ISBN: 0–521–78579–0.

[W+00] Edgar Wingender et al. TRANSFAC: an integrated system for gene expression regulation. *Nucleic Acids Res.*, 28:316–319, 2000.

[WCS97] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl*. O'Reilly, 1997. 3–930673–48–7.

[WF99] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 1999. ISBN: 1–558–60552–5.

[WH99] Clarke Wilson and Andrew S. Harvey. ClustalG: Software for analysis of activities and sequential events. In *IATUR Conference Proceedings*, 1999.

[Wue99] Andrew Wuensche. Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter. *Complexity*, 4(3):47–66, 1999.

Appendix A

MATERIALS AND METHODS

Operating System Microsoft Windows NT 4.0

Development was conducted on a PentiumIII/Microsoft Windows NT 4.0 workstation, on a Sun UltraEnterprise 450/Solaris 2.51, and on a PentiumII/RedHat Linux 6 cluster with 8 nodes.

Database POET Object Oriented Database Management system

POET Java developer license and object server version 5.0 and 5.1 for Windows NT from POET Software Corporation, 999 Baker Way — Suite 200, San Mateo, CA 94404 were used as the OODBMS.

There is a multitude of both academic and commercial OODBMS available. Since support and availablity of the database management system was critical for this work, the decision was made to use a commercial product. The database should be ODMG-compliant [Ce97] as much as possible and should be as mature a product as possible.

Because of the fast development, printed surveys on available database systems tend to become rapidly outdated. Several internet publications were reviewed instead [Sch98, Cha98, Bar98], leading to a preselection of Object Store, POET Object server and Objectivity. Out of these Object Store and POET were selected for evaluation, because both offered free evaluation licenses. Object Store was the market leader and POET is a partly Germany-based company, which would make support and contact easier.

Object Store did not support OQL queries. Since it was foreseeable that querying the data would be required, this would have meant implementing the query functionality in the application. I was not interested in inventing the wheel all over again, and the choice was made to use the POET database, which supports a subset of ODMG OQL.

Application Language Java

To make the code as portable as possible and ease creating a web-based interface Java Development Kit 1.16 through Java 2, Version 1.22 by Sun

Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA were used. Java is a modern object-oriented programming language that runs on a *virtual machine* instead of the actual hardware, making it platform independent as long as a virtual machine for the target platform exists. It fulfills all requirements mentioned in 1.3. Java can run as *applets* in a web browser, or as *servlets* in a web server, as long as they provide a virtual machine.

The development kit can be downloaded from `http://java.sun.com/j2se/`.

Scripting Perl - Practical Extraction and Report Language

For various data transformation and scripting tasks the the latest versions of ActiveState port of Perl[WCS97, SC97] were used. Perl has strong inbuilt pattern matching capabilities, which considerably ease processing of text files. Compared with shell scripts, it also has the advantage to work over different operating systems .

Perl is Open source distributed under the Perl artistic License. It can be downloaded from `http://www.perl.com/pub/language/info/software.html` or `http://www.activestate.com/Products/ActivePerl/`. Various additional packages for Perl and information can be obtained from the Comprehensive Perl Archive Network (CPAN) at `http://www.perl.com/CPAN-local/README.html` or one of its mirrors.

Web Interface HTML 4.01, XHTML Transitional, PaintShop Pro 4.01 and CorelDraw! 7.0, JavaScript and dynamically created pages

Static web pages were hand-written in HTML 4.01 [Rag00] and partly translated into XHTML 1.0 [P$^+$00] with the HTML Tidy [Rag00] utility. Dynamic effects for these static pages were created with JavaScript [Hei97, Jav00].

GIF images for pathway representation and web layout were created with CorelDraw! 7.0, available from Corel Corp., 1600 Carling Ave., Ottawa, Ontario K1Z 8R7, Canada `http://www.corel.com/` and with JASC Paint Shop Pro 4.01 `http://www.paintshoppro.com/`.

The dynamically created pages were produced by Java servlets that read data out of the database, processed it and formatted it as HTML for presentation.

Editors Symantec Visual Café, JPad, UltraEdit32, TextPad, Emacs

Java source code editing and testing was done with Integrated Development Environments (IDEs), first with Visal Café by Symantec Corporation (now distributed by WebGain `http://www.visualcafe.com/`), later with JPad available from Modelworks Software, 14612 NE 169th Street, Woodinville, WA 98072-6938, USA `http://www.jpadpro.com/`.

The composition of the document you are reading, some Java editing, HTML souce writing, Perl scripts and general all-purpose text file editing was done with UltraEdit32 by IDM Computer Solutions, Inc., 8209 Chestnut Hill Ct., West Chester, OH 45069, USA `http://www.ultraedit.com/`, with TextPad by Helios Software Solutions, PO Box 619, Longridge, PR3

2GW, England http://www.textpad.com/, and with GNU Emacs for Windows version 20.4 available under the GNU public license from GNU Software http://www.gnu.org/software/emacs/windows/ntemacs.html.

Webserver Apache/Jserv

The webservers used for the public presentation of the database are the Apache HTTP server Windows port by the the Apache Software Foundation available from http://ftp.cs.tu-berlin.de/pub/net/www/apache/ and the Apache JServ servlet server available from http://java.apache.org/, both distributed under their respective public licenses.

Versioning CVS

Source code versioning was done with the CVS open-source network-transparent version control system available from http://www.cvshome.org/CVS/index_html.

DDLab DDLab [Wue99] is an interactive graphics program for researching discrete dynamical networks. A network can be set up with any architecture from cellular automata to "random Boolean networks" The network may have heterogeneous neighborhood sizes. DDLab has been used for analysis of gene regulatory networks [S$^+$]. It is free for academic use and can be obtained from http://www.santafe.edu/~wuensch/ddlab.html.

Petri net UltraSAN

UltraSAN [MS93] is a simulation and analysis tool for Stochastic Activity Nets. It allows GUI-driven creation of models, performance analysis and simulation of these models. In UltraSAN time delays defined by a probability distribution can be specified for each activity. The tool had been used before for extracellular signaling cascades [ML97]. It is available at http://www.crhc.uiuc.edu/UltraSAN/UltraSAN.html.

UltraSAN [MS93] does not support a flatfile input format for model data. For this reason the data has to be entered manually using the GUI.

Graph layout Otter

Otter [Huf00] is a tool used for visualizing arbitrary network data that can be expressed as a set of nodes, links or paths. Otter was developed to handle visualization tasks for a wide variety of Internet data. It is available at http://www.caida.org/tools/visualization/otter/.

Typesetting LaTeX

This document was typeset with help of the TeX typesetting program by Donald E. Knuth [Knu97] using the LaTeX2e macro package and the free TeX distribution available from MiKTeX at http://www.miktex.org/.

Appendix B

ABBREVIATIONS

AC	adenylate cyclase
ACEDB	A C. Elegans Data Base
API	application programming interface
CSNDB	Cell Signaling Networks Data Base
DBMS	database management system
EC	enzyme classification
ER	endoplasmatic reticulum
HTML	hypertext markup language
HTTP	hypertext transfer protocoll
IP_3	inositol 1,4,5-triphosphate
MAP	mitogen activated kinase
NF-κB	Nuclear Factor κB
OODB	object-oriented database
OODBMS	object-oriented database management system
OQL	object query language
PI3K	phosphatidylinositol 3-kinase
PIP_2	phosphatidylinositol 4,5-bisphosphate
PKA	protein kinase, type A
SQL	standard query language
UML	unified modeling language
URL	uniform resource locator
WWW	world wide web
XML	extensible markup language

Appendix C

PATH ALIGNMENT

```
CLUSTAL G (1.1) multiple sequence alignment


IL-2@NF-kB|7                 ------------------------------------------MacweMaahmMaareMabzsMabzoMadhjMadrq-------------------------
IL-2@c-Myc|8                 ------------------------------------------MacweMaahmMaareMabzsMabzoMacbrMadugMaaba---------------------
IL-2@STAT5|3                 ------------------------------------------MacweMaahmMadki-----------------------------------------------
IL-2@Elk-1|11                ------------------------------------------MacweMaahmMaareMaazxMabjpMaclqMacxlMacxqMadmlMabtoMabtv-----
IL-2@p21|8                   ------------------------------------------MacweMaahmMaareMaazxMabjpMaclqMacxlMacns----------------------
IL-2@CREB2|7                 ------------------------------------------MacweMaahmMaareMaazxMabjpMadtzMaaao--------------------------
IL-2@c-Fos|7                 ------------------------------------------MacweMaahmMaareMaazxMabjpMadtzMacdy--------------------------
IL-2@AP-1|11                 ------------------------------------------MacweMaahmMaareMaazxMaabmMabskMadgwMacujMadxkMabbxMacry-----
IL-2@Jun|10                  ------------------------------------------MacweMaahmMaareMaazxMaabmMabskMadgwMacujMadxkMabbx----------
IL-3@NF-AT|7                 ------------------------------------------MacwdMacbrMaarjMabzsMacxsMabldMaaas--------------------------
PDGF@NF-AT|8                 ---------------------------------------MabbqMabsbMacbrMaarjMabzsMacxsMabldMaaas--------------------------
IL-9@NF-AT|9                 ---------------------------------------MacvwMaanuMadkmMacbrMaarjMabzsMacxsMabldMaaas--------------------------
IL-10@NF-AT|9                ---------------------------------------MaagoMadbqMadkmMacbrMaarjMabzsMacxsMabldMaaas--------------------------
IL-2@NF-AT|7                 ------------------------------------------MacweMaahmMaareMabzsMacxsMabldMaaas--------------------------
TNF-alpha_precursor_@NF-kB|5 ----------------------------------------------MacejMabfbMaaonMadffMadrq-----------------------------
TNF-alpha@NF-kB|4            ------------------------------------------------MabfbMaaonMadffMadrq-----------------------------
TNF-alpha_precursor_@NF-AT|6 ----------------------------------------------MacejMabfbMaaonMacusMabldMaaas--------------------------
TNF-alpha@NF-AT|5            ------------------------------------------------MabfbMaaonMacusMabldMaaas--------------------------
NGF@AP-1|8                   ------------------------------------------------------------MaajkMabenMadvoMabweMaapnMabtoMabbxMacry
NGF@Jun|7                    ------------------------------------------------------------MaajkMabenMadvoMabweMaapnMabtoMabbx-----
NGF@NF-kB|7                  ------------------------------------------------------------MaajkMabenMadvoMabweMaapnMabtoMadrq-----
NGF@Elk-1|7                  ------------------------------------------------------------MaajkMabenMadvoMabweMaapnMabtoMabtv-----
prostaglandin_D2@PPAR-gamma|3 ---------------------------------------MaarbMaazyMacbq-----------------------------------------------
IL-4@STAT6|3                 -----MacwcMabsuMadkh-----------------------------------------------------------------------------------
L-glutamate@CREB|4           ----------------------------MabmpMaddrMaalrMadqx-------------------------------------------------------
PDGF@CREB|6                  --------------------MabbqMabsbMacenMaddrMaalrMadqx-------------------------------------------------------
NRG-2@CREB|4                 ----------------------------MadbeMaddrMaalrMadqx-------------------------------------------------------
serotonin@NF-kB|6            ---------------------------------MadekMacblMacfuMacrlMadxkMadrq-------------------------------------------
serotonin@Elk-1|9            ---------------------------------MadekMacblMacfuMacrlMadxkMacxqMadmlMabtoMabtv-----------------------
serotonin@AP-1|7             ---------------------------------MadekMacblMacfuMacrlMadxkMabbxMacry-------------------------------------
serotonin@Jun|6              ---------------------------------MadekMacblMacfuMacrlMadxkMabbx-----------------------------------------
serotonin@NF-AT|9            ---------------------------------MadekMacblMacfuMacrlMaaejMacxlMacxsMabldMaaas--------------------------
serotonin@p21|7              ---------------------------------MadekMacblMacfuMacrlMaaejMacxlMacns----------------------------------
serotonin@c-Myc|9            ---------------------------------MadekMacblMacfuMacrlMaaejMacxlMacbrMadugMaaba-----------------------
IL-12@STAT4|3                --------------------MaagnMaazwMadkk-------------------------------------------------------------------
IL-3@NF-kB|7                 ------------------------------------------MacwdMacbrMaarjMabzsMabzoMadhjMadrq--------------------------
PDGF@NF-kB|8                 ---------------------------------------MabbqMabsbMacbrMaarjMabzsMabzoMadhjMadrq--------------------------
IL-3@c-Myc|4                 ------------------------------------------MacwdMacbrMadugMaaba---------------------------------------
PDGF@c-Myc|5                 ---------------------------------------MabbqMabsbMacbrMadugMaaba---------------------------------------
IL-3@AP-1|9                  ------------------------------------------MacwdMacbrMaclqMacxlMacxqMadmlMabtoMabbxMacry--------------
IL-3@Jun|8                   ------------------------------------------MacwdMacbrMaclqMacxlMacxqMadmlMabtoMabbx--------------------
IL-3@Elk-1|8                 ------------------------------------------MacwdMacbrMaclqMacxlMacxqMadmlMabtoMabtv--------------------
IL-3@p21|5                   ------------------------------------------MacwdMacbrMaclqMacxlMacns----------------------------------
PDGF@AP-1|10                 ---------------------------------------MabbqMabsbMacbrMaclqMacxlMacxqMadmlMabtoMabbxMacry--------------
PDGF@Jun|9                   ---------------------------------------MabbqMabsbMacbrMaclqMacxlMacxqMadmlMabtoMabbx--------------------
PDGF@Elk-1|9                 ---------------------------------------MabbqMabsbMacbrMaclqMacxlMacxqMadmlMabtoMabtv--------------------
PDGF@p21|6                   ---------------------------------------MabbqMabsbMacbrMaclqMacxlMacns----------------------------------
IL-9@STAT3|3                 ---------------------------------MacvwMaanuMadkm-------------------------------------------------------
IL-9@NF-kB|9                 ---------------------------------MacvwMaanuMadkmMacbrMaarjMabzsMabzoMadhjMadrq--------------------------
IL-9@c-Myc|6                 ---------------------------------MacvwMaanuMadkmMacbrMadugMaaba---------------------------------------
IL-9@AP-1|11                 ---------------------------------MacvwMaanuMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabbxMacry--------------
IL-9@Jun|10                  ---------------------------------MacvwMaanuMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabbx--------------------
IL-9@Elk-1|10                ---------------------------------MacvwMaanuMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabtv--------------------
IL-9@p21|7                   ---------------------------------MacvwMaanuMadkmMacbrMaclqMacxlMacns----------------------------------
IL-10@STAT3|3                ---------------------------------MaagoMadbqMadkm-------------------------------------------------------
IL-10@NF-kB|9                ---------------------------------MaagoMadbqMadkmMacbrMaarjMabzsMabzoMadhjMadrq--------------------------
IL-10@c-Myc|6                ---------------------------------MaagoMadbqMadkmMacbrMadugMaaba---------------------------------------
IL-10@AP-1|11                ---------------------------------MaagoMadbqMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabbxMacry--------------
IL-10@Jun|10                 ---------------------------------MaagoMadbqMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabbx--------------------
IL-10@Elk-1|10               ---------------------------------MaagoMadbqMadkmMacbrMaclqMacxlMacxqMadmlMabtoMabtv--------------------
IL-10@p21|7                  ---------------------------------MaagoMadbqMadkmMacbrMaclqMacxlMacns----------------------------------
IL-10@STAT1|3                ---------------------------------MaagoMadbqMadko-------------------------------------------------------
acetylcholine@CREB|7         -----MaaezMadtyMadigMabswMaaloMaafdMadqx-----------------------------------------------------------------
acetylcholine@CREB1|7        -----MaaezMadtyMadigMabswMaaloMaafdMaaaq-----------------------------------------------------------------
acetylcholine@NF-kB|10       -----MaaezMadtyMadigMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------------
acetylcholine@AP-1|11        -----MaaezMadtyMadigMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------------
acetylcholine@Jun|10         -----MaaezMadtyMadigMabswMaaloMaafdMadgwMacujMadxkMabbx-----------------------------------------------
```

```
adrenaline@CREB|7          -----MaavqMacoxMadigMabswMaaloMaafdMadqx-----------------------------------------------------------
adrenaline@CREB1|7         -----MaavqMacoxMadigMabswMaaloMaafdMaaaq-----------------------------------------------------------
adrenaline@NF-kB|10        -----MaavqMacoxMadigMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------
adrenaline@AP-1|11         -----MaavqMacoxMadigMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------
adrenaline@Jun|10          -----MaavqMacoxMadigMabswMaaloMaafdMadgwMacujMadxkMabbx-------------------------------------------
adrenaline@CREB2|7         -----MaavqMacoxMadigMabswMaaloMadtzMaaao-----------------------------------------------------------
adrenaline@c-Fos|7         -----MaavqMacoxMadigMabswMaaloMadtzMacdy-----------------------------------------------------------
acetylcholine@CREB2|7      -----MaaezMadtyMadigMabswMaaloMadtzMaaao-----------------------------------------------------------
ACTH@NF-kB|10              -----MacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------
CRH@NF-kB|11               MadnaMacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------
ACTH@AP-1|11               -----MacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------
CRH@AP-1|12                MadnaMacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------
ACTH@Jun|10                -----MacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMabbx-------------------------------------------
CRH@Jun|11                 MadnaMacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMabbx-------------------------------------------
ACTH@Elk-1|13              -----MacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMacxqMadmlMabtoMabtv---------------------------
CRH@Elk-1|14               MadnaMacckMacqpMaafqMabswMaaloMaafdMadgwMacujMadxkMacxqMadmlMabtoMabtv---------------------------
ACTH@CREB|7                -----MacckMacqpMaafqMabswMaaloMaafdMadqx-----------------------------------------------------------
CRH@CREB|8                 MadnaMacckMacqpMaafqMabswMaaloMaafdMadqx-----------------------------------------------------------
ACTH@CREB1|7               -----MacckMacqpMaafqMabswMaaloMaafdMaaaq-----------------------------------------------------------
CRH@CREB1|8                MadnaMacckMacqpMaafqMabswMaaloMaafdMaaaq-----------------------------------------------------------
ACTH@NF-AT|11              -----MacckMacqpMaafqMabswMaaloMaafdMadgwMacujMabauMaajhMaaas-------------------------------------
CRH@NF-AT|12               MadnaMacckMacqpMaafqMabswMaaloMaafdMadgwMacujMabauMaajhMaaas-------------------------------------
ACTH@CREB2|7               -----MacckMacqpMaafqMabswMaaloMadtzMaaao-----------------------------------------------------------
CRH@CREB2|8                MadnaMacckMacqpMaafqMabswMaaloMadtzMaaao-----------------------------------------------------------
ACTH@c-Fos|7               -----MacckMacqpMaafqMabswMaaloMadtzMacdy-----------------------------------------------------------
CRH@c-Fos|8                MadnaMacckMacqpMaafqMabswMaaloMadtzMacdy-----------------------------------------------------------
GH-RH@AP-1|11              -----MachzMadkxMaafqMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------
GH-RH@Jun|10               -----MachzMadkxMaafqMabswMaaloMaafdMadgwMacujMadxkMabbx-------------------------------------------
GH-RH@NF-kB|10             -----MachzMadkxMaafqMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------
GH-RH@Elk-1|13             -----MachzMadkxMaafqMabswMaaloMaafdMadgwMacujMadxkMacxqMadmlMabtoMabtv---------------------------
GH-RH@CREB|7               -----MachzMadkxMaafqMabswMaaloMaafdMadqx-----------------------------------------------------------
GH-RH@NF-AT|11             -----MachzMadkxMaafqMabswMaaloMaafdMadgwMacujMabauMaajhMaaas-------------------------------------
GH-RH@CREB1|7              -----MachzMadkxMaafqMabswMaaloMaafdMaaaq-----------------------------------------------------------
GH-RH@CREB2|7              -----MachzMadkxMaafqMabswMaaloMadtzMaaao-----------------------------------------------------------
GH-RH@c-Fos|7              -----MachzMadkxMaafqMabswMaaloMadtzMacdy-----------------------------------------------------------
serotonin@CREB|6           ----------MadekMabwiMabswMaaloMaafdMadqx-----------------------------------------------------------
serotonin@CREB1|6          ----------MadekMabwiMabswMaaloMaafdMaaaq-----------------------------------------------------------
serotonin@CREB2|6          ----------MadekMabwiMabswMaaloMadtzMaaao-----------------------------------------------------------
serotonin@c-Fos|6          ----------MadekMabwiMabswMaaloMadtzMacdy-----------------------------------------------------------
adenosine@NF-kB|9          ----------MaahbMaadwMabswMaaloMaafdMadgwMacujMadxkMadrq-------------------------------------------
adenosine@Elk-1|12         ----------MaahbMaadwMabswMaaloMaafdMadgwMacujMadxkMacxqMadmlMabtoMabtv---------------------------
adenosine@AP-1|10          ----------MaahbMaadwMabswMaaloMaafdMadgwMacujMadxkMabbxMacry---------------------------------------
adenosine@Jun|9            ----------MaahbMaadwMabswMaaloMaafdMadgwMacujMadxkMabbx-------------------------------------------
adenosine@CREB|6           ----------MaahbMaadwMabswMaaloMaafdMadqx-----------------------------------------------------------
adenosine@NF-AT|10         ----------MaahbMaadwMabswMaaloMaafdMadgwMacujMabauMaajhMaaas-------------------------------------
adenosine@CREB1|6          ----------MaahbMaadwMabswMaaloMaafdMaaaq-----------------------------------------------------------
adenosine@CREB2|6          ----------MaahbMaadwMabswMaaloMadtzMaaao-----------------------------------------------------------
adenosine@c-Fos|6          ----------MaahbMaadwMabswMaaloMadtzMacdy-----------------------------------------------------------
acetylcholine@c-Fos|3      -----MaaezMadtyMacdy-------------------------------------------------------------------------------
acetylcholine@Elk-1|11     ----------------------------MaaezMadtyMadigMaaffMacktMaclqMacxlMacxqMadmlMabtoMabtv---------------
adrenaline@Elk-1|11        ----------------------------MaavqMacoxMadigMaaffMacktMaclqMacxlMacxqMadmlMabtoMabtv---------------
acetylcholine@NF-AT|10     ----------------------------MaaezMadtyMadigMaaffMacktMaclqMacxlMacxsMabldMaaas---------------------
acetylcholine@p21|8        ----------------------------MaaezMadtyMadigMaaffMacktMaclqMacxlMacns-------------------------------
acetylcholine@c-Myc|10     ----------------------------MaaezMadtyMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
adrenaline@NF-AT|10        ----------------------------MaavqMacoxMadigMaaffMacktMaclqMacxlMacxsMabldMaaas---------------------
adrenaline@p21|8           ----------------------------MaavqMacoxMadigMaaffMacktMaclqMacxlMacns-------------------------------
adrenaline@c-Myc|10        ----------------------------MaavqMacoxMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
ACTH@c-Myc|15              -----MacckMacqpMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
CRH@c-Myc|16               MadnaMacckMacqpMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
ACTH@p21|13                -----MacckMacqpMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacns-------------------------------
CRH@p21|14                 MadnaMacckMacqpMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacns-------------------------------
adenosine@c-Myc|14         ----------MaahbMaadwMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
adenosine@p21|12           ----------MaahbMaadwMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacns-------------------------------
GH-RH@c-Myc|15             -----MachzMadkxMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacbrMadugMaaba-----------------------
GH-RH@p21|13               -----MachzMadkxMaafqMabswMaaloMaafdMabynMadigMaaffMacktMaclqMacxlMacns-------------------------------
EGF@AP-1|10                ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacxqMadmlMabtoMabbxMacry---------------
EGF@Jun|9                  ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacxqMadmlMabtoMabbx------------------
EGF@NF-kB|9                ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacxqMadmlMabtoMadrq------------------
EGF@Elk-1|9                ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacxqMadmlMabtoMabtv------------------
EGF@NF-AT|8                ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacxsMabldMaaas---------------------
EGF@p21|6                  ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacns-------------------------------
EGF@c-Myc|8                ---------------------------------------MaczjMadbwMabjpMaclqMacxlMacbrMadugMaaba-----------------------
EGF@CREB2|5                ---------------------------------------MaczjMadbwMabjpMadtzMaaao-----------------------------------
EGF@c-Fos|5                ---------------------------------------MaczjMadbwMabjpMadtzMacdy-----------------------------------
IL-1@NF-kB|7               --------------------MacwfMabhkMacunMaaisMabtgMadlaMadrq-------------------------------------------
                                               .    :    :    .                         .    .
```

```
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/CREB2
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/calmodulin/calcineurin/NF-AT
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc
ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/p21
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/CREB2
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/calmodulin/calcineurin/NF-AT
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc: 16
CRH/ACTH/ACTH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/p21
```

EGF/EGF receptor/Grb2/MAPK cascade/CREB2
EGF/EGF receptor/Grb2/MAPK cascade/c-Fos
EGF/EGF receptor/Grb2/Sos/Ras/PI3-kinase/PKB/c-Myc
EGF/EGF receptor/Grb2/Sos/Ras/Rac/JNK/NF-AT
EGF/EGF receptor/Grb2/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
EGF/EGF receptor/Grb2/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun
EGF/EGF receptor/Grb2/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun/AP-1
EGF/EGF receptor/Grb2/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/NF-kB
EGF/EGF receptor/Grb2/Sos/Ras/p21
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/CREB2
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/CREB
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/calmodulin/calcineurin/NF-AT
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc
GH-RH/GH-RH receptor/G-alpha-s/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/p21
IL-10/IL-10 receptor/STAT1
IL-10/IL-10 receptor/STAT3
IL-10/IL-10 receptor/STAT3/PI3-kinase/PKB/c-Myc
IL-10/IL-10 receptor/STAT3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac-1/reactive oxygen species/NF-kB
IL-10/IL-10 receptor/STAT3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac/JNK/NF-AT
IL-10/IL-10 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
IL-10/IL-10 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun
IL-10/IL-10 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun/AP-1
IL-10/IL-10 receptor/STAT3/PI3-kinase/Sos/Ras/p21
IL-12/IL-12 receptor/STAT4
IL-1/IL-1 receptor/TRAF6/NIK/IKK/I-kB/NF-kB
IL-2/IL-2 receptor/Lck/LAT/Grb2/MAPK cascade/CREB2
IL-2/IL-2 receptor/Lck/LAT/Grb2/MAPK cascade/c-Fos
IL-2/IL-2 receptor/Lck/LAT/Grb2/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
IL-2/IL-2 receptor/Lck/LAT/Grb2/Sos/Ras/p21
IL-2/IL-2 receptor/Lck/LAT/PLC-gamma1/IP3/IP3 receptor/Ca(II)/protein kinase C/Jun
IL-2/IL-2 receptor/Lck/LAT/PLC-gamma1/IP3/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
IL-2/IL-2 receptor/Lck/VAV/Rac-1/PI3-kinase/PKB/c-Myc
IL-2/IL-2 receptor/Lck/VAV/Rac-1/reactive oxygen species/NF-kB
IL-2/IL-2 receptor/Lck/VAV/Rac/JNK/NF-AT
IL-2/IL-2 receptor/STAT5
IL-3/PI3-kinase/PKB/c-Myc
IL-3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac-1/reactive oxygen species/NF-kB
IL-3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac/JNK/NF-AT
IL-3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
IL-3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun
IL-3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun/AP-1
IL-3/PI3-kinase/Sos/Ras/p21
IL-4/IL-4 receptor/STAT6
IL-9/IL-9 receptor/STAT3
IL-9/IL-9 receptor/STAT3/PI3-kinase/PKB/c-Myc
IL-9/IL-9 receptor/STAT3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac-1/reactive oxygen species/NF-kB
IL-9/IL-9 receptor/STAT3/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac/JNK/NF-AT
IL-9/IL-9 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
IL-9/IL-9 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun
IL-9/IL-9 receptor/STAT3/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun/AP-1
IL-9/IL-9 receptor/STAT3/PI3-kinase/Sos/Ras/p21
L-glutamate/NMDA receptor/CaMKIV/CREB
NGF/CREB
NGF/NGF receptor/C3G/Rap1/B-Raf/MAP-kinase/Elk-1
NGF/NGF receptor/C3G/Rap1/B-Raf/MAP-kinase/Jun
NGF/NGF receptor/C3G/Rap1/B-Raf/MAP-kinase/Jun/AP-1
NGF/NGF receptor/C3G/Rap1/B-Raf/MAP-kinase/NF-kB
NRG-2/NMDA receptor/CaMKIV/CREB
PDGF/PDGF receptor/Fyn/NMDA receptor/CaMKIV/CREB
PDGF/PDGF receptor/PI3-kinase/PKB/c-Myc
PDGF/PDGF receptor/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac-1/reactive oxygen species/NF-kB
PDGF/PDGF receptor/PI3-kinase/PtdIns(3,4,5)P3/VAV/Rac/JNK/NF-AT
PDGF/PDGF receptor/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
PDGF/PDGF receptor/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun
PDGF/PDGF receptor/PI3-kinase/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Jun/AP-1
PDGF/PDGF receptor/PI3-kinase/Sos/Ras/p21
SRPK1/SR proteins
TNF-alpha(precursor)/TNF-alpha/TNF receptor1/DR3/NF-kB
TNF-alpha(precursor)/TNF-alpha/TNF receptor1/TRAF2/JNK/NF-AT
TNF-alpha/TNF receptor1/DR3/NF-kB
TNF-alpha/TNF receptor1/TRAF2/JNK/NF-AT
WSL-1/NF-kB
acetylcholine/muscarinic acetylcholine receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc
acetylcholine/muscarinic acetylcholine receptor/Gi/G-gamma/Src/Sos/Ras/Rac/JNK/NF-AT
acetylcholine/muscarinic acetylcholine receptor/Gi/G-gamma/Src/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
acetylcholine/muscarinic acetylcholine receptor/Gi/G-gamma/Src/Sos/Ras/p21
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/MAPK cascade/CREB2
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/CREB
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
acetylcholine/muscarinic acetylcholine receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
acetylcholine/muscarinic acetylcholine receptor/c-Fos
activin/FAST-1
adenosine/A2b receptor/adenylyl cyclase/cAMP/MAPK cascade/CREB2
adenosine/A2b receptor/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/CREB
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/calmodulin/calcineurin/NF-AT
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1

adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc
adenosine/A2b receptor/adenylyl cyclase/cAMP/Protein Kinase A/beta-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/p21
adrenaline/alpha2-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/PI3-kinase/PKB/c-Myc
adrenaline/alpha2-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/Rac/JNK/NF-AT
adrenaline/alpha2-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
adrenaline/alpha2-adrenergic receptor/Gi/G-gamma/Src/Sos/Ras/p21
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/MAPK cascade/CREB2
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/CREB
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/Jun/AP-1
adrenaline/alpha2-adrenergic receptor/Gi/adenylyl cyclase/cAMP/Protein Kinase A/IP3 receptor/Ca(II)/protein kinase C/NF-kB
cortisol/glucocorticoid receptor
estradiol/AP-1
estradiol/ERE
estradiol/estrogen receptor
glucocorticoid/glucocorticoid receptor
progesterone/progesterone receptor
prostaglandin D2/15d-PGJ2/PPAR-gamma
retinoic acid/p21
retinoic acid/retinoic acid receptor
serotonin/5-HT2CR/phospholipase C/diacylglycerol/GNRP/Ras/PI3-kinase/PKB/c-Myc
serotonin/5-HT2CR/phospholipase C/diacylglycerol/GNRP/Ras/Rac/JNK/NF-AT
serotonin/5-HT2CR/phospholipase C/diacylglycerol/GNRP/Ras/p21
serotonin/5-HT2CR/phospholipase C/diacylglycerol/protein kinase C/Jun
serotonin/5-HT2CR/phospholipase C/diacylglycerol/protein kinase C/Jun/AP-1
serotonin/5-HT2CR/phospholipase C/diacylglycerol/protein kinase C/NF-kB
serotonin/5-HT2CR/phospholipase C/diacylglycerol/protein kinase C/Raf/MAP-kinase-kinase/MAP-kinase/Elk-1
serotonin/serotonin receptor/adenylyl cyclase/cAMP/MAPK cascade/CREB2
serotonin/serotonin receptor/adenylyl cyclase/cAMP/MAPK cascade/c-Fos
serotonin/serotonin receptor/adenylyl cyclase/cAMP/Protein Kinase A/CREB1
serotonin/serotonin receptor/adenylyl cyclase/cAMP/Protein Kinase A/CREB
thyroxine/thyroxine receptor

Appendix D

PETRI NET PARAMETERS

Table D.1: Non-zero or Variable Initial Markings for SAN Model *A_EGFR*

Place	***Marking***
EGF	*GLOBAL_S(egf)*
EGFR	*166*
EGF_EGFR	*GLOBAL_S(egf_egfr)*
GDP_Ras	*GLOBAL_S(gdp_ras)*
GRB2	*1000*
GTP_Ras	*GLOBAL_S(gtp_ras)*
MAPK_X	*GLOBAL_S(mapk_x)*
SHC	*500*
SOS	*100*

Table D.2: Activity Time Distributions for SAN Model *A_EGFR*

Activity	***Distribution***	***Parameter values***
A1	exponential	
	rate	$MARK(EGF) *$ $MARK(EGFR) *$ 0.007
A10	exponential	
	rate	$MARK(SOS\_X) *$ $MARK(GRB2) *$ 0.000041667
A10B	exponential	
	rate	$0.0168 *$ $MARK(SOS\_X\_GRB2)$
A1B	exponential	
	rate	$0.25 *$ $MARK(EGF\_EGFR)$
A2	exponential	
	rate	$0.002 *$ $MARK(EGF\_EGFR)$
A2B	exponential	
	rate	$MARK(EGF\_EGFR\_INTERNAL) *$ 0.00033
A3	exponential	
	rate	$0.2 *$ $MARK(EGF\_EGFR) *$ $MARK(SHC) /$ $(833 + MARK(SHC))$
A4	exponential	
	rate	$0.0016667 *$ $MARK(SHC\_X)$
A5	exponential	
	rate	$MARK(SOS) *$ $MARK(GRB2) *$ 0.000041667
A5B	exponential	
	rate	$0.0168 *$ $MARK(SOS\_GRB2)$
A6	exponential	

Table D.3: Activity Time Distributions for SAN Model *A_EGFR*

Activity	***Distribution***	***Parameter values***
	rate	$MARK(SHC\_X) *$ $MARK(SOS\_GRB2) *$ *0.000833*
A6B	exponential	
	rate	*0.1* $*$ $MARK(SHC\_X\_SOS\_GRB2)$
A7	exponential	
	rate	*0.02* $*$ $MARK(SHC\_X\_SOS\_GRB2) *$ $MARK(GDP\_Ras)$ / $(505 + MARK(GDP\_Ras))$
A8	exponential	
	rate	$MARK(SOS\_X) *$ *0.001*
A9	exponential	
	rate	$10 * MARK(MAPK\_X) *$ $MARK(SOS)$ / $(2564 + MARK(SOS))$

Table D.4: Non-zero or Variable Initial Markings for SAN Model *H_MAPK*

Place	***Marking***
MAPK	*360*
MAPKK	*180*
MAPK_X	$GLOBAL\_S(mapk\_x)$
MKP1	*3*
PKC	$GLOBAL\_S(pkc)$
PP2_A	*224*
Raf	*200*

Table D.5: Activity Time Distributions for SAN Model *H_MAPK*

Activity	***Distribution***	***Parameter values***
H1	exponential	
	rate	$4.0 * MARK(PKC) * MARK(Raf) / (66666.0 + MARK(Raf))$
H10	exponential	
	rate	$0.15 * MARK(MAPKK\_XX) * MARK(MAPK) / (46.0 + MARK(MAPK))$
H11	exponential	
	rate	$0.15 * MARK(MAPKK\_XX) * MARK(MAPKtyr\_X) / (46.0 + MARK(MAPKtyr\_X))$
H12	exponential	
	rate	$1.0 * MARK(MKP1) * MARK(MAPKtyr\_X) / (66.0 + MARK(MAPKtyr\_X))$
H13	exponential	
	rate	$1.0 * MARK(MKP1) * MARK(MAPK\_X) / (66.0 + MARK(MAPK\_X))$
H2	exponential	
	rate	$10.0 * MARK(MAPK\_X) * MARK(Raf\_X) / (25641.0 + MARK(Raf\_X))$
H3	exponential	
	rate	$6.0 * MARK(PP2\_A) * MARK(Raf\_X) / (15656.0 * MARK(Raf\_X))$
H4	exponential	
	rate	$6.0 * MARK(PP2\_A) * MARK(Raf\_XX) / (15656.0 * MARK(Raf\_XX))$
H5	exponential	
	rate	$0.04 * MARK(GTP\_Ras) * MARK(Raf\_X)$

Table D.6: Activity Time Distributions for SAN Model *H_MAPK*

Activity	***Distribution***	***Parameter values***
H5B	exponential	
	rate	*0.5 ** *MARK(GTP_Ras_Raf_X)*
H6	exponential	
	rate	*0.105 ** *MARK(GTP_Ras_Raf_X) ** *MARK(MAPKK) /* *(159.0 + MARK(MAPKK))*
H7	exponential	
	rate	*0.105 ** *MARK(GTP_Ras_Raf_X) ** *MARK(MAPKK_X) /* *(159.0 + MARK(MAPKK_X))*
H8	exponential	
	rate	*6.0 * MARK(PP2_A) ** *MARK(MAPKK_X) /* *(15656.0 + MARK(MAPKK_X))*
H9	exponential	
	rate	*6.0 * MARK(PP2_A) ** *MARK(MAPKK_XX) /* *(15656.0 + MARK(PP2_A))*

Table D.7: Activity Time Distributions for SAN Model *O_CLOCK*

Activity	***Distribution***	***Parameter values***
Tick	deterministic	
	value	*1*

Table D.8: Output Gate Definitions for SAN Model *O_CLOCK*

Gate	***Definition***
limit	*if (MARK(Clock) > GLOBAL_S(stimulus_time)) {* *MARK(Clock) = 0;* *} else {* *MARK(Clock) = MARK(Clock) + 2;* *MARK(EGF) = GLOBAL_S(egf);* *}*

Appendix E

ADMINISTRATIVA AND THANKS

First an foremost my gratefulness goes to Dr. Edgar Wingender, who made this work possible. His scientific brilliance, his encouragement, his open-mindedness in discussions and his support for my toils made all the difference. His integrity, generosity, initiative and humor as a person are elevating and unforgettable to me.

Thanks to my family and to Katja Schroeder who loved and supported me in every possible way.

Admiration and thanks go to my referees Prof. Ursula Bilitewski and Prof. Hans-Dieter Ehrich, who appreciated this interdisciplinary work, and were always supportive and encouraging.

I want to thank all of my great collegues from GBF, Biobase and Roche, who made working and living in Braunschweig and Palo Alto a pleasure.

From GBF: Dr. Ines Liebich, for her tolerance and patience on working with me, Annegret Bischoff for administrative help and discussions on social injustice, Dr. Ingmar Reuter for nightshift work companionship and helping me with the pains of user-, database- and web server administration, Dr. Alexander and Dr. Olga Kel and Xin Chen for sharing their culture, Thorsten Gross, Maik Christensen, Michael Ledwa and Volker Drewes for an informatician's approach and for working with and porting the database and creating clients, Dr. Anatolij Potapov and Dr. Klaus Seidel for discussions of the biological and modeling aspects and help with user support, Rolf Gohla for Unix hints, Dr. Holger Michael and Dr. Thomas Heynemeyer for helpful comments and interest.

From Biobase: Dr. Volker Matys for lunch companionship and interest in language, Holger Karas for relaxedness and more knowledge than he'd admit, Tanja Roggenbuck for pragmatism and arranging travel support, Mathias Krull, Susanne Pistor, Sandra Urbach, Claudia Choi and Ulrike Götze for entering data and turning the idea of Transpath into a usable knowledge base, Ellen Fricke for morbidity, Manuela Prüeßfor sarcasm, Dagmar Karas for workouts. And thank you to all the other wonderful colleagues of Biobase, Andreas Adler, Axel Wagner, Antje Wingender, Anja Diel, Richard Ohnhäuser, Robert Geffers, Susanne Thiede and Jochen "PGP" Striepe.

From Roche: Dr. Michael Liebman for making it possible for me to go there,

Kay Coen, for doing a wonderful job arranging things and driving me around, big thanks to Dr. Lei Du, Eugene, and Randall for enlightening discussions and friendly help, to Eugene for setting up the Linux Cluster and to Nigel, Jason and Ashok for setting up the PC, to Keith, Donglei, Yuan-Yuan and Debbie for sharing their spare time with me.

Thanks go to the many fine scientists out there who sacrificed some of their time to discuss all kinds of questions with me: Dr. Adam Arkin, Drew Endy, Dr. Stefan Schuster, Dr. Andrew Wuensche, Dr. Clarke Wilson, Dr. Ralf Zimmer and Dr. Ronald Somogyj. Thanks to the users of Transpath for making it all worthwhile, especially the ones who pointed out mistakes or had ideas for improvement.

A final thank-you goes to Dr. Takako Igarashi-Takai, who provided the dataset of her CSNDB database for early experiments and integration into the Transpath viewer and had time for many interesting discussions.